NOVELTY DETECTION + COEVOLUTION = AUTOMATIC PROBLEM DECOMPOSITION: A FRAMEWORK FOR SCALABLE GENETIC PROGRAMMING CLASSIFIERS

by

Andrew R. McIntyre

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
November 2007

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "NOVELTY DETECTION + COEVOLUTION = AUTOMATIC PROBLEM DECOMPOSITION: A FRAMEWORK FOR SCALABLE GENETIC PROGRAMMING CLASSIFIERS" by Andrew R. McIntyre in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Dated: November 9, 2007

External Examiner: _____

Research Supervisor: _____
M. I. Heywood

Examining Committee: _____
E. E. Milios

_____
S. S. R. Abidi

_____

_____

DALHOUSIE UNIVERSITY

DATE: November 9, 2007

AUTHOR: Andrew R. McIntyre

TITLE: NOVELTY DETECTION + COEVOLUTION = AUTOMATIC PROBLEM DECOMPOSITION: A FRAMEWORK FOR SCALABLE GENETIC PROGRAMMING CLASSIFIERS

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: Ph.D. CONVOCATION: May YEAR: 2008

Table of Contents

List of Tables

List of Figures

Abstract

A novel approach to the classification of large and unbalanced multi-class data sets is presented where the widely acknowledged issues of scalability, solution transparency, and problem decomposition are addressed simultaneously within the context of the Genetic Programming (GP) paradigm. A cooperative coevolutionary training environment that employs multi-objective evaluation provides the basis for problem decomposition and reduced solution complexity. Scalability is achieved through a Pareto competitive coevolutionary framework, allowing the system to be readily applied to large data sets without recourse to hardware-specific speedups. A key departure from the canonical GP approach to classification involves expressing the output of GP in terms of a non-binary, local membership function (Gaussian), where it is no longer necessary for an expression to represent an entire class. Decomposition is then achieved through reformulating the classification problem as one of cluster consistency, where individuals learn to associate subsets of training exemplars with each cluster. Classification problems are now solved by several specialist classifiers as opposed to a single 'super' individual. Finally, although multi-objective methods have been reported previously for GP classification domains, we explicitly formulate the objectives for cooperative behavior. Without this the user is left to choose a single individual as the overall solution from a front of solutions. This work is able to utilize the entire front of solutions without recourse to heuristics for selecting one individual over another or duplicating behaviors between different classifiers.

Extensive benchmarking is performed against related frameworks for classification including Genetic Programming, Neural Networks, and deterministic methods. In contrast to classifiers evolved using competitive coevolution alone, we demonstrate the ability of the proposed coevolutionary model to provide a non-overlapping decomposition or association between learners and exemplars, while returning statistically significant improvements in classifier performance. In the case of the Neural Network methods, benchmarking is conducted against the more challenging second order neural learning algorithm of conjugate gradient optimization (previous comparisons limit Neural Network training to first order methods). The ensuing comparison indicated that most data sets benefit from the proposed algorithm, which remains competitive even against the well-established deterministic algorithms, such as C4.5.

List of Abbreviations Used

ADF	Automatically Defined Functions
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
AUC	Area Under the Curve
BNF	Backus-Naur Form
CBD	Communal Binary Decomposition
CFG	Context Free Grammar
CGM	Conjugate Gradient Method
CMGE	Competitive Multi-objective Grammatical Evolution
DecMO-GP	Decomposed Multi-objective Genetic Programming
DELPHI	Dominance Evaluation of Learners on Pareto Hillclimbing Individuals
DM	Decision Maker
DR	Detection Rate
DRS	Dynamic Range Selection
DSS	Dynamic Subset Selection
EC	Evolutionary Computation
EMO	Evolutionary Multi-objective Optimization
FN	False Negative
FOCUS	Find Only and Compete Undominated Sets
FP	False Positive
FPGA	Field Programmable Gate Array
FPR	False Positive Rate
GA	Genetic Algorithm
GE	Grammatical Evolution
GP	Genetic Programming
IPCA	Incremental Pareto Coevolution Archive
KDD	Knowledge Discovery in Databases

LEX	Lexical Analyzer
LMF	Local Membership Function
LOOCV	Leave One Out Cross Validation
LP	Linear Perceptron
ML	Machine Learning
MLP	Multi-layer Perceptron
MO	Multi-objective
MOGA	Multi-objective Genetic Algorithm
MOGE	Multi-objective Optimization with Grammatical Evolution
MV	Majority Voting
MVGPC	Majority Voting Genetic Programming Classifier
NB	Naive Bayes
NFL	No Free Lunch
NN	Neural Network
NSGA	Nondominated Sorting Genetic Algorithm
OET	Orthogonal Evolution of Teams
PCGA	Pareto Converging Genetic Algorithm
PGPC	Pareto-coevolutionary Genetic Programming Classifier
PM	Probabilistic Multi-class
POPE-GP	Pseudo Objective Parsimony Enforcement Genetic Programming
RAM	Random Access Memory
ROC	Receiver Operating Characteristic
RSS	Random Subset Selection
SIMD	Single Instruction Multiple Data
SPEA	Strength Pareto Evolutionary Algorithm
SRS	Static Range Selection
SS	Steady State
SSE	Sum of Squared Error
TD	Training Data
TN	True Negative
TP	True Positive

TS Training Set

UCI University of California Irvine

VEGA Vector Evaluated Genetic Algorithm

WTA Winner Take All

YACC Yet Another Compiler Compiler

Acknowledgements

I would first like to thank my supervisor, Dr. Malcolm Heywood, for his unwavering support and kind direction throughout this process; I cannot begin to adequately express my gratitude for his enlightening guidance and commitment to my academic development. I would also like to thank my committee members, Dr. Evangelos Milios, Dr. Raza Abidi, along with my external, Dr. Una-May O'Reilly, for their thorough, insightful questions and invaluable feedback.

I would also like to thank my family, especially my Mother and Father (Connie and Gary) for their enduring encouragement, patience and understanding. They are my greatest role models and inspiration.

Finally I would like to gratefully acknowledge the support received over the course of my graduate career, including: Nova Scotia Health Research Foundation (NSHRF), Pre-Competitive Advanced Research Network (PRECARN), Mathematics of Information Technology and Complex systems (MITACS), Canada Foundation for Innovation (CFI) New Opportunities and the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery programs, as well as the industrial funding received through the Telecom Applications Research Alliance (TARA) and SwissCom Innovations AG (Switzerland).

Chapter 1

Introduction

Advances in digital computers and mass storage systems have triggered an acceleration in the global acquisition, processing and warehousing of data in countless forms. With communication networks now providing widespread inter-connectivity and spurring yet new forms of data collection, considerable interest is being taken in the challenging problem of finding meaningful predictive and/or descriptive models of these resources. Machine Learning (ML) concerns the automation of this process through computer-driven model optimization according to performance criteria using past examples or experience [2]. Of principal interest to this thesis is the ML task known as 'classifier learning from examples', or simply *classification*. Classification represents one of the most active areas of ML research, having significant potential for predictive applications in medical, biological, technological, economic and linguistic domains, to name a few [32] [29] [94] [108] [114] [118]. This thesis is explicitly concerned with the development of a robust classification framework under the Evolutionary Computation (EC) paradigm of Machine Learning, with specific emphasis on the Genetic Programming (GP) algorithm.

This chapter establishes the classification context of Machine Learning and introduces the EC paradigm, discussing both motivational aspects as well as practical design considerations relevant to the application of GP to classification. A high-level introduction to GP is presented to initiate the unfamiliar reader with the basic algorithm and associated terminology. Next a short discussion of the proposed framework is provided along with the specific GP and classification objectives to be addressed. The chapter concludes with an organizational summary of the remainder of the document.

1.1 Machine Learning

Distinctions are typically made between the three primary domains of ML, specifically: supervised, unsupervised and reinforcement learning. Supervised learning refers to the use of labeled data where each input vector X is accompanied by a target output (or label) Y. An input vector X (also referred to as an *exemplar, case*, or *pattern*) consists of pre-defined descriptive elements called *features* or *attributes*. When the labels (Y) takes on discrete values, the supervised learning task is called *classification* and involves discovering procedures for categorizing cases into classes, whereas continuous values for Y are associated with the *regression* task, or *function modeling*. The unsupervised context employs data in the absence of associated labels and generally involves descriptive modeling of data, where the primary objective is to find regularities or groupings – a process that is also known as *density estimation*. Reinforcement learning builds *policies* or sequences of actions to take in order to reach a pre-specified goal by assessing previously appropriate sequences and generating updated policies. This thesis is mainly concerned with the classification context of the supervised learning domain.

1.2 Design Issues for Machine Learning

Irrespective of the learning task, successful application of ML methods to a diverse range of problems hinges on careful consideration of the key design issues, which effectively incorporate the practitioner's *a priori* knowledge and domain-specific assumptions into the ML task. Specifically, there are three fundamental considerations to be addressed when designing (or applying) a machine learning framework [6]:

Representation : Defines the form of the candidate solutions by establishing a set of elements (an *alphabet*) from which a model may be induced. The elemental arrangements (and/or permissible configurations thereof) define the model's *free parameters* which are to be optimized by the machine learning algorithm. The set of all possible model combinations under a given representation is referred to as the *representation space*. Moreover, the representation may enforce many

constraints among elements (thereby reducing the free parameters) or, alternatively, permit a greater degree of elemental flexibility (increasing the number of free parameters). An appropriate choice of representation is generally problem dependent, requiring *a priori* knowledge of the target domain with respect to the various constraint tradeoffs and their corresponding effects on the other design considerations, discussed below.

Cost function : Specifies a computationally tractable estimate for quantitative characterization of model behavior. Specifically, the cost function provides a performance map of the representation space (defined above) to a *search space*. A more tightly constrained representation attempts to minimize the ambiguity in this mapping, whereas additional freedom in the representation can result in a more complex interaction between the representation and search spaces. In conjunction with the cost function, 'stop criteria' are typically defined which specify the halting point for the model search. Stop criteria are generally defined in terms of a computational limit (e.g., a maximum number of credit assignment steps – in the case of ML this may correspond to an upper limit on the number of models to constructed through an iterative process of refinement) or a satisfactory level of performance (often in terms of the cost function), where further 'improvement' could be considered negligible or entirely undesirable.

Credit assignment : Provides a mechanism to guide the algorithm toward promising candidate solutions by relating the information about the search space performance (provided by the cost function) back to the representation space.

These design considerations critically influence an algorithm's ability to search, evaluate and express solutions thereby introducing both implicit and explicit assumptions about the problem domain. This influence is commonly referred to as *inductive bias*, which is defined as any criterion (aside from consistency with the data) used to favor one candidate solution (or *hypothesis*) over another [38]. Inductive bias is common to all learning algorithms, with certain biases having more suitability for some application domains as opposed to others. The two main types of inductive bias are introduced below with a brief discussion of their relationship with the aforementioned

ML design issues.

1.2.1 Representational Bias

Representational bias is firstly influenced by how an instance of the data is described in terms of the input features. This might include implications for descriptive specifications according to the use of numeric vectors, nominal or binary features, etc. Representational bias may also refer to the implicit nature of the solution representation, or what is known as the solution *representation language* (e.g., trees, rules, networks and so forth). Representational bias may therefore provide certain advantages (or impose particular constraints) according to:

Element design : The basic elements or building blocks of any machine learning model are pre-specified by the representation. An example might include building blocks suggested by biological models, such as neural synapses [48], or alternatively, the problem might be considered in terms of facilitating a mapping process between the original input space, to the output (classification) space. In the latter case, *a priori* decisions are made with the selection of a 'representative' set of operators necessary to perform the mapping in much the same way that instruction sets are designed for computing systems [49]. In either case, domain knowledge is used to place limits on the representational flexibility with the basic computing elements.

Model structure : Elements used to compose or configure a model might be deployed with very tight rules for model building (as in the full connectivity pattern in artificial Neural Network models) or require learning of relationships between elements. The latter, for example, results in the concept of 'syntactic closure' under Genetic Programming; that is, the output from any model building element must be appropriate for the input to any element [61].

Decisions made regarding a representational bias will naturally impact the complexity of the resulting learning task, as well as influencing the ability to make decisions appropriate to the domain in question. Examples include support for linear versus non-linear model building, full connectivity versus sparse connectivity (of basic model

building elements), and support for encouraging reuse of structures previously discovered as appropriate for problem solving. Moreover, depending on the elements selected, the ensuing model may either be readily interpretable (highly descriptive), or take the form of a black box solution. In short, the bias assumed when declaring the building blocks for an ML model can have a considerable impact on many facets of the ensuing solutions, as well as interacting with design decisions made regarding the credit assignment process and establishing appropriate cost functions. In the case of Genetic Programming, for example, we may potentially assert that the set of elemental building blocks be Turing Complete, thus capable of a wide range of problem solving; however, this says nothing of the ability to find a suitably 'good' solution. Indeed, assuming such a representation may make the process of identifying useful solutions more difficult as the search space can become artificially large.

1.2.2 Search Bias

Being that the search space of most practical problems is generally too large and hence computationally prohibitive to enumerate directly, a search heuristic is responsible for finding 'optimal' solutions. The two latter ML design considerations introduced in Section 1.2 (namely, cost function specification and credit assignment) together help define how the representation space will be explored and exploited in the search for 'optimal' regions of the search space. This influence is referred to as *search bias*. An example might include 'smoothness' constraints on an algorithm's cost function, which requires a differentiable relationship between free parameters and cost function such that gradient information can be employed to direct the optimization procedure. In such a case, a tightly constrained representation results in an informative, minimally ambiguous mapping between representation and search space that will typically be accompanied by a local credit assignment mechanism, capable of exploiting the locally relevant feedback supplied by the cost function. This might include a greedy approach to credit assignment, which favors pursuit of the local model change(s) that will provide the most incremental improvement relative to the current model. This 'local exploitation' behavior is typically desired but trades off in the exploration behavior, as local optima can impede progress toward globally 'optimal' solutions.

In contrast, an ML approach having more freedom among representational elements (typically resulting in more complex interactions between representation space and search space) does not necessarily rely on meaningful local feedback from the cost function in the credit assignment approach driving the ensuing search. Stochastic credit assignment algorithms, for example, have no specific greedy constraints on search direction and are free to make random or probabilistic selections during the optimization process. While this class of algorithms has the potential to provide more search exploration (potentially returning different solutions on each initialization) and avoid convergence on locally optimal regions of the search space, these tradeoffs are typically realized at the cost of algorithmic complexity. Stochastic search is the primary focus of the current work, specifically Genetic Programming (GP) algorithms of the Evolutionary Computation (EC) paradigm. The primary contribution of this thesis will be to provide algorithmic advances that attempt to improve search efficiency in terms of credit assignment, without stepping outside of the stochastic search domain.

1.2.3 No Free Lunch

All machine learning algorithms employ design choices that hold inherent representation and search biases having direct implications for the search processes and ensuing solutions. Different learning algorithms therefore hold certain advantages and disadvantages according to the problem. The 'No Free Lunch' (NFL) theorems formalize this, noting that any elevated performance by one algorithm over one class of problems is exactly paid for in performance over another class [119]. Given that there exists no 'universal best' approach, good machine learning performance is generally contingent on the appropriate matching of algorithms with problems [118].

1.3 Evolutionary Computation

Evolutionary Computation is a branch of Machine Learning concerned with algorithms that employ metaphors from natural evolution to model search processes for optimization. Among these algorithms exist assorted differences in terms of approach, most notably with respect to solution representation and credit assignment (in the

form of *population* and *search* operators) [5]. EC algorithms differ significantly from most ML models (stochastic or greedy) on account of their flexibility in terms of representation and credit assignment. Moreover, these algorithms make explicit use of a multipoint search that attempts to address the credit assignment process by building new models from more than one model in the current 'population' of candidate solutions. The procedures are motivated by the neo-Darwinian principle of survival of the fittest, where transitions between search states are stochastic in nature (favoring the sampling of 'fit' candidate solutions) and as such enable the algorithm to make different 'decisions' under the same environmental conditions. Such a property has the potential to afford robust exploration of the search space and aid in the escape from local optima.

As opposed to the classical search methods, EC algorithms permit user-specification of the representation and cost function, making comparatively few assumptions about the nature of the underlying problem. While such properties have the potential to offer significant advantages over the greedy search procedures, the complexity issue remains widely acknowledged and provides one of the principal motivations for the current work.

1.4 Genetic Programming

Genetic Programming (GP) is an EC learning paradigm that attempts to evolve a population of computer programs that evolve towards a common goal as they interact with a training environment [6]. GP is an attractive approach to learning that provides a stochastic global search requiring little in the way of *a priori* knowledge and allows for flexible representations and problem-specific evaluation schemes, making it readily applicable to a wide range of problem domains [61]. In a classification configuration, GP provides a stochastic training process that returns diverse, analytic models while making no assumptions about the distribution of the underlying data and requiring minimal pre-processing, particularly with respect to scaling and feature selection of the training data [75].

The standard Genetic Programming algorithm is motivated by a neo-Darwinian model of organic evolution, the process responsible for the emergence of complex and

well-adapted structures in nature [5]. Of specific significance to the GP model is the Darwinian notion of 'survival of the fittest'. In basic terms, Darwin posits that well-adapted individuals within a population hold a natural survival (and therefore selectional) advantage that drives the eventual propagation of above-average genetic material through reproduction. Under the evolutionary metaphor, GP candidate solutions are known as *individuals*. The low level, genetic representation of an individual is referred to as the *genotype, genome*, or *chromosome* and consists of elemental genetic units known as *genes*, each being an instance of a particular *allele* [86] [6]. The expression of the genetic material in terms of observable characteristics is known as the *phenotype*. The basic relationship between the genotype and phenotype is indicated in Figure 1.1, where two items are of specific interest. The first being the duality between the representation (or decision) space and search (or objective) space, which correspond to the population genotype and phenotype representations, respectively. The second being the association of genotype and phenotype representations with the genetic and population operators, respectively. The critical link between the two is fitness evaluation in terms of the problem objective(s), which influences the ensuing credit assignment.

1.4.1 Problem Design

GP design choices implicitly bias the landscape of the search space and, by extension, the efficiency with which solutions are found. The practitioner therefore faces numerous decisions in terms of design and simulation details that require special considerations between GP algorithms and problem types. These are matters of considerable interest within the research community and a large volume of the literature aims to address the key elements, including: models of solution representation and operator specification, formulation of cost function with respect to objectives, credit assignment mechanisms and parameterization. These will be discussed in the following sections.

1.4.2 Representation

The standard canonical algorithm for GP, summarized in Algorithm 1, employs a population P of individuals which represent candidate solutions in the form of models, or basic computer programs. Solution representations are highly dependent upon the variant of GP under use and can be tailored according to problem domain and practical experience.

Among EC practitioners, most would recognize the canonical model to consist of a fixed-length l, binary string-based genotype representation $g = \{0,1\}^l$, with phenotype p being the result of a mapping function, m, taking g to a suitable output corresponding to a candidate solution, $p = m(g)$. The preference for a binary string representation is derived from the original *schema theorem* [5], however the work of Antonisse and more recently Michalewicz demonstrated that integer representations are also entirely appropriate and quite often beneficial [4] [84]. Koza's canonical GP algorithm, however, employs a strictly tree-based representation where both genotype and phenotype are variable length parse-trees of LISP programs, or S-expressions, which can be interpreted directly or with modest processing [60] [61]. Recently, the indirect binary string genotype representation has been established under the GP context, perhaps most notably in the cases of Linear GP [90] [39] and Grammatical Evolution (or GE) [91] where the phenotype corresponds to state-machine programs or intermediate grammatical rule selectors, respectively. The latter is of specific relevance to this thesis.

Under the conventional representation, the GP practitioner is responsible for supplying a representation in terms of a language framework that is appropriate and reasonably efficient for solving the problem under consideration. A typical language specification will consist of terminals and non-terminals (e.g., operators or functions) that will be used to compose the expression (computer program in the case of GP) of an individual. In the traditional GP formulation, for example, the practitioner is specifically responsible for providing the *terminal* and *functional* sets, which Koza defines as the 'alphabet' of the programs to be evolved. The terminal set consists of variables, inputs and constants, while the function set may contain specific operators (e.g., arithmetic operators: $+$, -, $\div$, $\times$; procedural instructions: move left, move

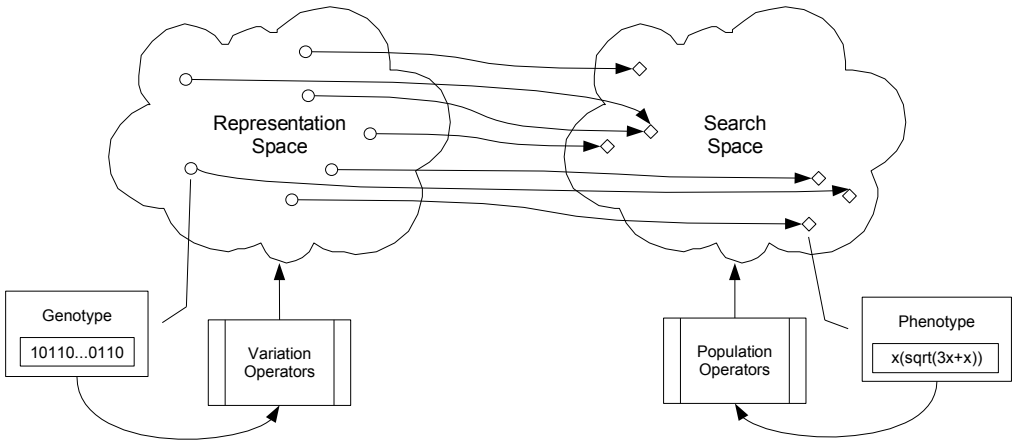


Figure 1.1: Genotype / Phenotype Relationship.

right, speed up, slow down and so forth) to be carried out (e.g., in a simulation environment) in conjunction with the terminals or other more complicated functions [61]. Moreover, the language specification must be consistent (genotype to phenotype) and provide closure under the variation operators, meaning that all permissible modifications to a program in the language must result in a syntactically valid program as defined by the language framework.

1.4.3 Credit Assignment: Exploration vs. Exploitation

Choice of parameters such as population size and evolutionary run length directly affect the search characteristics of the GP algorithm. For example, large populations run over a small number of generations approximate a random walk through the search space, while a small population run for a large number of generations can simulate a hill-climbing or local search [76]. Under each of these configurations the GP algorithm demonstrates conflicting search characteristics, which are advantageous when combined yet potentially ineffective on their own, introducing the classical credit assignment tradeoff known as *exploration versus exploitation*.

Exploration of the search space is both desired and required in the GP algorithm; in particular, the exploration features allow the GP to sample a diverse range of solutions and facilitate a means of stochastic escape from local optima. When promising

regions of the search space are encountered, however, there must be an exploitative pursuit of the corresponding genetic material (i.e., ability to perform a local or 'hill-climbing' search) to investigate the potential for global optima. That is to say, while a stochastic credit assignment that enables good exploration is desired, its effects must be counterbalanced with exploitation behavior in order to avoid an overly randomized search that is unable to actively discover optimal solutions. Likewise in defining the exploitation behavior – while it provides the means to actively direct a local search, it lacks the capacity to avoid convergence in local basins of attraction (i.e., the stochastic escape readily provided by exploitation behavior).

To summarize, the GP algorithm is able to provide a wide ranging global search when it is maximally explorative; however, this style of search falls into direct conflict with the exploitation objective, which requires a focusing or hill-climbing capacity that can be largely absent in the exploration behavior and vice versa. When adequately combined, the elements of exploration and exploration can provide a strong degree of consistency and efficiency in the search process; striking an appropriate balance or tradeoff between these aspects is therefore a primary goal in GP. Moreover, this tradeoff represents a recurring theme in the design and practical implementation of any GP, in that many of algorithm's features can be seen as contributing to the exploration and/or exploitation modes of search.

1.4.4 Operators

Operators simulate the natural selection, variation and inheritance of genetic material and are applied iteratively throughout the evolutionary run. Operators provide the primary transitional mechanisms of the GP algorithm, and are specifically responsible for guiding the population toward optimal regions of the search space through selection, variation and replacement; that is to say, credit assignment.

In the GP context, operators are broadly described as either population or genetic (variation) operators. The population operators are problem independent and are typically applied stochastically across the search space according to the fitness distribution of the population. Specifically, these operators define processes for selection of parents and replacement of population members, where both processes are

biased to favor the more fit individuals (on average). In contrast, genetic variation operators are applicable to points (genotypes) in the representation space and are used to stochastically recombine and alter genetic material.

GP operators explore the representation space and exploit individuals that have advantageous characteristics in the search space; i.e., the candidate solutions are sampled from the search space, whereas the search itself is conducted in the representation space.

Selection and Replacement

The selection and replacement operators are responsible for choosing among individuals upon which to base successive population instances. Each process is biased in favor of individuals exhibiting advantageous features (i.e., above average fitness); the magnitude of this bias is referred to as the *selection pressure*. Two schemes have become particularly well established within the GP community and these will be discussed in detail below:

1. Fitness-proportionate selection with generational replacement;

2. Tournament-based selection with steady state (SS) replacement.

The fitness proportionate selection context is often associated with the generational replacement form of GP; that is, performance in terms of fitness is used to proportionately bias the selection across the population P, such that an individual $i \in P$ having fitness $f(i)$ has a probability for selection P_s specified by:

$$P_s(i) = \frac{f(i)}{\sum_{j \in P} f(j)} \tag{1.1}$$

This can be readily implemented as a biased roulette wheel simulation where each individual is allocated a number of slots on the wheel according to P_s. On each spin, the individual associated with the winning slot is selected as a parent. In the generational algorithm, parent pairs produce offspring by way of probabilistic application of the variation operators, which are then copied to a second population, P^*. When the number of individuals in P^* reaches the original population size,

P^* replaces P. Iterating this process, the generational GP algorithm replaces its populations with successive generations of new (and hopefully improved) individuals.

The denominator in Equation 1.1 requires the calculation of a global statistic in the fitness proportionate approach. A less computationally intensive alternative is the tournament-based selection process, which is frequently paired with a steady state replacement strategy. Tournament-based selection samples the population for a uniform random subset of individuals to participate in a fixed tournament size, $s = 4k$, with $k \in \mathbb{Z}^+$ and $k << \frac{|P|}{4}$ which provides the basis for the parent selection process. The choice of s provides a direct means to influence selection pressure, with larger values of s enforcing more pressure. Moreover, a tournament may be run for selection of each parent or the most fit pair may be selected as parents from a single tournament. The replacement occurs following the production of offspring through stochastic application of the variation operators where offspring deterministically replace the worst tournament performers. The steady state approach therefore employs no centralized mechanism for explicitly buiding 'generations' and as such the population remains in a 'steady state' of update (i.e., the operators are continually modifying the population).

The evolutionary properties resulting from the two approaches have been demonstrated to be distinct [113]; in particular, the generational model has a much lower 'turn over' rate than the SS tournament model. That is to say, the generational model will let weaker individuals linger in the population much longer than the SS tournament. Moreover, the SS tournament is explicitly elitist, which has been shown to provide advantageous convergence properties over the generational model without an additional elitist operator [101].

Genetic Variation

Variation operators are responsible for the recombination and modification of genetic material in the representation space. Two widely used operators are of specific relevance to this work: crossover and mutation. The design of these operators may be specific to the GP representation being employed; however, the general approach will be discussed here.

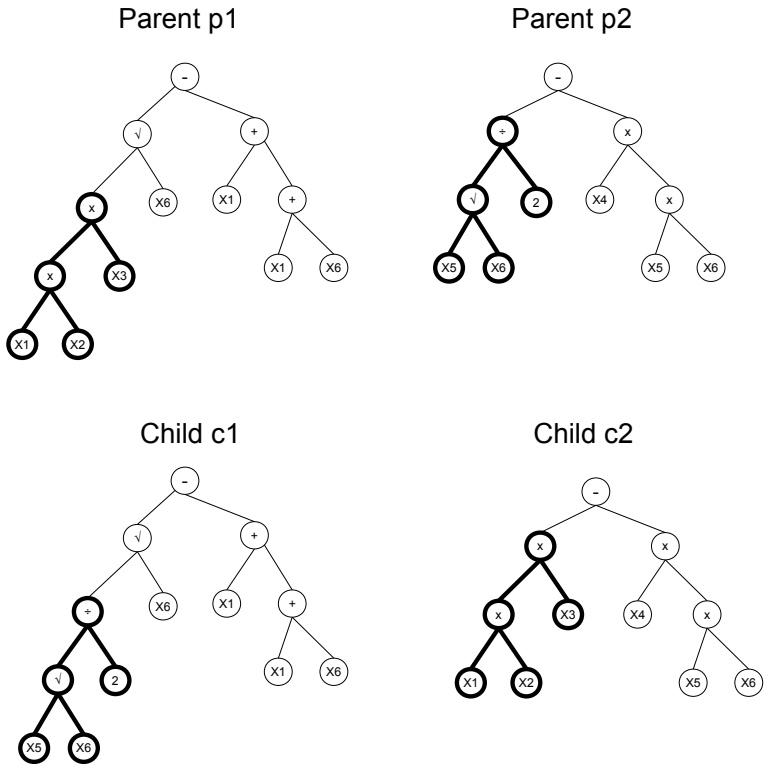


Figure 1.2: Crossover operator in traditional GP as applied to parents p1, p2 resulting in offspring c1, c2. Child c2 now contains the expression for evaluating the difference in volume between two boxes $((x_1 \times x_2 \times x_3) - (x_4 \times x_5 \times x_6))$.

The crossover operator plays a primarily recombinatorial role in GP and is applied to parent pairs stochastically with a probability of application P_{xo}. The outcome of the application of the crossover operator is an exchange of genetic material, with the goal being an exchange corresponding to advantageous material, or *building blocks* between parents that is inherited by offspring. When applied in the most common form (single point crossover) the crossover operator selects a point in the genotype of each parent and swaps the genetic material beyond these points between parents (Figure 1.2). The crossover points are traditionally chosen based on a uniform random selection, constrained by representational suitability. When the test for the application of crossover fails, an exact copy of parent material is assigned to each offspring in lieu of recombination or additional tests for the application of mutation.

The mutation operator is responsible for genetic variation and is applied to each offspring stochastically with a probability of application P_m. The mutation operator is representationally dependent (i.e., it must obey the closure properties of chosen representation) and involves applying suitable random alterations to the genotype of the offspring (Figure 1.3). Such an operator is either applied gene-wise or individual-wise where the latter is combined with an additional test to establish the particular instruction (and/or field) affected.

1.4.5 Problem Objectives and Definition of Cost Function

Evaluation of individuals at the phenotype level and the corresponding fitness assignment (according to a user-defined cost function) play critical roles in GP; success is highly dependent on the appropriate formulation of problem objectives. As such, cost functions may be evaluated directly or according to a function of the problem under consideration. Typically objectives are to be maximized or minimized and a problem may consist one or many objectives (see Section 2.3).

Unlike other machine learning methods, GP provides considerable flexibility in the design of the cost (fitness) function. Neural networks and kernel methods require differentiable cost functions, whereas decision trees enforce a greedy entropy minimization cost function. Thus the potential exists for making the fitness function more closely reflect the desired goals of the problem domain relative to other machine

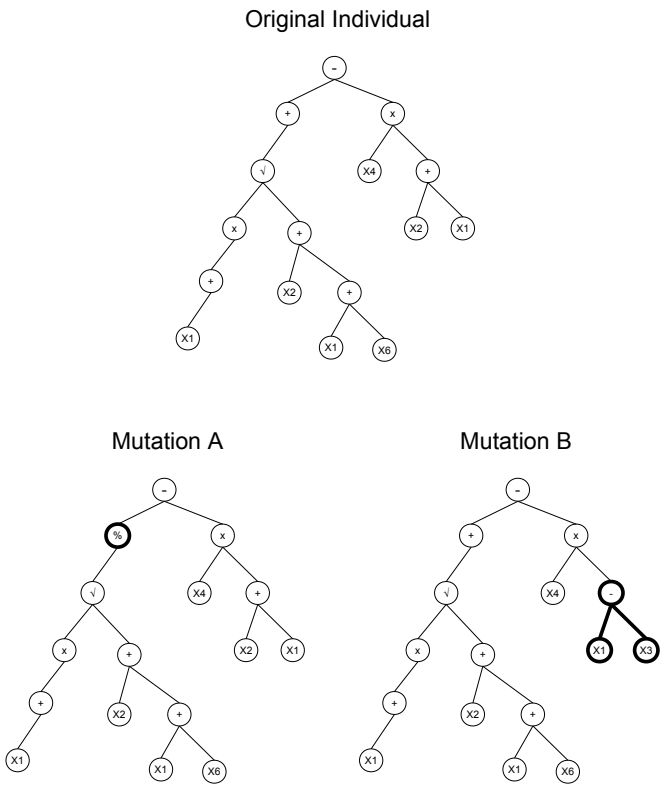


Figure 1.3: Mutation operator in traditional GP. Mutation A: single node replacement; Mutation B: subtree replacement.

learning methods. Moreover, the utility of the multi-objective fitness functions is readily achieved, further extending the flexibility of machine learning methods as a whole [55].

Under the supervised learning context (the primary domain of this thesis) training data in the form of exemplars or fitness cases are supplied as input / output pairs. In the single objective case, the training of GP proceeds as outlined in Algorithm 1 and the evaluation of an individual can be a relatively straightforward procedure such as counting the number of *hits*; that is, fitness cases (exemplars) that satisfy a predefined minimum threshold of error. Under the classification domain, output labels generally take the form of integers denoting class membership of the corresponding (typically multi-dimensional) input. For example, the hit-based fitness function popularized by Koza for classification takes the form of a count of the number of matching labels provided by an individual [61] [62]. To this end, Koza employed a *wrapper* or *activation function* to map the GP output from a 1-dimensional number line into a binary space, representing in and out-of-class labels (Koza did not consider the multi-class case). We will later show that such a methodology has a significant impact on the ensuing behavior of the classifier (e.g., novelty detection versus discrimination, single versus multiple solutions, weak learner or ensemble methods versus non-overlapping problem decomposition).

Under the single individual, single objective context the fitness evaluation processes can be relatively transparent; however the introduction of multiple objectives or multiple models per solution can introduce new problems in fitness evaluation and the ensuing credit assignment. Of particular relevance to this thesis is the multi-objective credit assignment problem and how to determine fitness in the face of several factors contributing to the degree of fulfillment of the stated objective(s). The problem has specific interest when contributing influences conflict in relation to the objective(s). This will be discussed in detail for the multi-objective case in Section 2.3 and the multi-model case in Section 2.4.

1.5 GP Classification and Computational Overhead

The canonical GP algorithm for the classification context was established by Koza

Algorithm 1 gpMain - Canonical model for GP.

Input: Population of programs P, training data TD.
Output: Best program, bp.

```
1: Initialize P randomly
2: Evaluate P over TD
3: Assign fitness to P
4: while ! Stop criteria do
5:    Select parents from P, biased for fitness
6:    Apply variation operators to produce children C
7:    Evaluate C over TD
8:    Assign fitness to C
9:    Insert C into P
10: end while
11: bp := Program in P having maximum fitness
```

[60] [61] and is outlined in Algorithm 1. The supervised evaluation process involves calculating the fitness of the population as a whole (establishing the basis for credit assignment) before application of the selection and variation operators. Such a process implies that the *a priori* declared cost function (typically taking the form of a count of correctly classified training exemplars) be evaluated over all training exemplars. Such a process creates a computationally expensive inner loop at line 7 of Algorithm 1 and renders the basic GP algorithm prohibitively costly for large training sets. Specifically, the number of program evaluations necessary to evolve a solution under the canonical GP algorithm can be estimated as a function of the number of classes c, number of initializations I, number of generations G, population size P, and size of training data $|TD|$ [61], Equation 1.2. Typical values for each of the four GP run parameters are summarized in Table 1.1 with the corresponding upper and lower bounds on the number of evaluations. One of the goals of this thesis is to present algorithms that decouple the number of evaluations actually performed from the values assigned *a priori* to these four generic GP run parameters.

$$\#Evals = c \times I \times G \times P \times |TD| \tag{1.2}$$

Clearly the number of classes c and the size of the training data $|TD|$ will be fixed for any given problem. These two factors are traditionally addressed through

Table 1.1: Typical GP parameter values for estimating total evaluations.

Term	**Lower**	**Upper**
c	2	9
I	50	100
G	50	500
P	50	1000
$\|TD\|$	100	500000
$\#Evals$	2.50E+007	2.25E+014

hardware specific speedups (e.g., multiple concurrent initializations spread over a large number of nodes) and will be discussed in greater detail in Section 2.5.2. Selection of I, G, and P is something of an art, with Koza, for example, taking the philosophical approach that the search process be driven through reproduction and crossover. Such a model requires a comparatively large population of 'genetic material' from which to compose solutions. An alternative approach frequently employed under the linear GP representation is to introduce mutation to maintain diversity under a smaller population model.

1.6 Approach and Objectives

The current work considers the evolution of classifiers comprised of multiple, cooperating models with Evolutionary Multi-objective Optimization (EMO) providing a basis for comparing the performance of individuals in the presence of multiple performance objectives using the criterion of Pareto dominance [93]. We assume that candidate solutions are performing a mapping from input space X, to a one dimensional (classification) output space. However, in contrast to the global wrapper typically assumed, a local membership function (LMF – a Gaussian) is employed within a cooperative context to encourage collaborations, providing the 'novelty detection' basis for automatic problem decomposition. The result is a set of mappings effectively responding to different subsets of the original input space and mapping them to different (non-overlapping) output spaces. Moreover, the well known issue regarding the scalability of GP with respect to data set size is addressed through the

use of a Pareto competitive coevolution training framework which provides multi-class solutions in a single run of GP. Specifically, the current research establishes the Competitive Multi-objective Grammatical Evolution (CMGE) framework for classification. CMGE represents a Pareto coevolutionary version of the Multi-objective Grammatical Evolution (MOGE) framework [82] that is capable of addressing the problems that typically preclude a GP approach to classification, including:

Scalability : The inability of GP in its canonical form to address problems involving large numbers of training exemplars (tens of thousands or millions; see Equation 1.2) is well documented in the literature [108] [22] [6]. This problem is directly addressed through the novel use and reformulation of the Incremental Pareto Coevolution Archive (IPCA) algorithm [23]. In contrast to previous approaches we do not rely on hardware-related factors, opting rather to sub-sample the data set and use competitive coevolution to retain the most useful learners and exemplars.

Class Imbalance : Poor representation of minority classes in the set of training exemplars can lead GP individuals to focus on the majority classes in an attempt to maximize fitness. Previously, this has been addressed through niching and fitness sharing methods; however, these methods required careful selection and tuning of parameters to ensure niche maintenance and to minimize sensitivity to genetic drift [80] [85]. In this thesis, we will address the problem through the design of a coevolutionary training model that explicitly requires a balanced representation of patterns from majority and minority classes. Such an approach is enforced by the growing literature on machine learning under unbalanced data sets. The work of [116] makes a particularly strong case for the balanced sampling of major and minor classes.

Solution Transparency : Simple solutions have the capacity to provoke further analysis into factors contributing to the various classes of data presented. Without this requirement, GP is free to evolve expressions that are highly complex and inefficient [6] [61], leading to computational overheads during training and

producing final solutions that are difficult to analyze directly for underlying significance post-training. This issue will be addressed through the multi-objective (MO) framework. Again, use is made of the Pareto MO model in which node count and error are simultaneously minimized producing a Pareto front of solutions representing the trade off between the two [25] [93]. Such a scheme tends to be superior to comprising the cost function from the weighted combination of error and complexity terms or lexicographic (hierarchical) formulations in which complexity is employed as a 'tie breaker' during parent selection [77], as both approaches assume a fixed *a priori* weighting of one factor relative to the other, the significance of which has unknown impact on the solutions found.

Problem Decomposition : The tendency for the GP population to converge on a single 'super individual' that is required to classify the entire data set with a single expression is well established [6], yet may be undesirable in light of the solution transparency objective. Moreover, the ability of the system to decompose the problem into a series of small, simple solutions provides the basis for modularity of the classifier. This issue will be addressed through the evolution of local membership functions that reformulate the classification problem as one of cluster consistency in combination with a unique approach to forming the objectives of the Pareto MO framework. In particular we are able to reformulate the classification problem as one of locating mappings from the input space to a one-dimensional output space such that class-consistent clusters are rewarded. We do not require all exemplars from the same class to be mapped to the same cluster, but explicitly reward the combination of individuals who successfully classify all exemplars as a group. Such an approach makes no assumption regarding the number of individuals comprising a solution, and is applicable to binary and multi-class domains. Related approaches in which Pareto MO GP formulations have been proposed for multi-class classification are limited to single GP individuals representing each class [121] [93] and there is no problem decomposition beyond different individuals (from the same population) representing different classes.

Multi-class Applications : Problems involving more than two outputs for each exemplar (in vs. out-of-class, i.e., binary classification) will be handled by a single run of the proposed system, whereas the usual GP approach to such problems involves configuring each class as a separately evolved binary classifier, requiring N runs of GP for an N-class problem [58]. This will be accomplished through an efficient design of the training algorithm, where the evaluation of cluster-consistency dictates class allocations and the stopping criteria ensures sufficient training with respect to all classes. That is to say, evolution of multi-class classifiers takes place in parallel with a single training run providing individuals of all classes. Deployment of the resulting classifiers takes place in parallel where this requires stronger performance of the classifiers as a whole than the hierarchical case, which can assign classifiers with best false positive rates to the first layers, masking the comparatively poor performance of classifiers appearing later in the hierarchy. The principal benefit of the parallel classification model is that we are now able to explicitly address the multi-label classification domains in which exemplars might be a member of multiple classes.

The principal objectives of this research are therefore to address these issues for classification within the context of GP by combining the MOGE algorithm with a novel coevolutionary training model that simultaneously performs competitive and cooperative coevolution. That is to say, the competitive model decouples the fitness evaluations and scales the approach to large data sets, where as cooperative coevolution facilitates problem decomposition.

1.7 Thesis Overview

Chapter 2 introduces the background literature relevant to Genetic Programming, particularly with respect to the design of GP systems under the classification context. Specifically, this chapter begins by describing the GP framework used in this thesis (Grammatical Evolution, or GE) along with the related design tradeoffs relevant to representations and search operators under this paradigm. We next present a survey of the binary and multi-class approaches from the recent GP literature focusing on the various treatments of the traditional pathologies of GP associated with

scalability, class imbalance, solution transparency, problem decomposition and multi-class applications as introduced above. We introduce the notion of a multi-objective cooperative coevolutionary framework and discuss the formulation of classification-specific objectives for GP as well as the notion of 'opponents' (training patterns) as objectives.

Chapter 3 details the algorithms pertaining to the framework presented in this thesis and describes the relevant design choices in light of the traditional pathologies of GP under classification. Specifically, underlying motivations are discussed for the use of EMO to evolve the desired class-consistent cooperation and parsimony behaviors and the use of IPCA to build multi-class teams while supporting a balanced, scalable representation of the training data. The chapter concludes with a detailed computational complexity analysis.

Chapter 4 describes the benchmarking methodology, including the comparative systems, data sets and evaluation procedures. We provide motivation for the approach taken and support the benchmark selections, providing references to related benchmark literature while discussing some recently reported results. Moreover, we introduce a separate methodology for stochastic and deterministic comparisons, where the latter require special treatment in order to frame single-run experiments against multiple initialization experiments synonymous with GP.

Chapter 5 begins with the comparative results of the current framework with the canonical and scalable variants of GP. These results highlight the classical pathologies related to GP under the classification task and demonstrate the capacity for these issues to be addressed by the current framework. Results are indicated for classification performance, training times and solution complexities with the proposed CMGE framework demonstrating results that are statistically significant in their superiority in the vast majority of cases. These results motivate further comparisons of the CMGE framework and a baseline, 'scalable' variant of standard GP (RssGE). The chapter concludes with a brief discussion of the classification results of an alternative Pareto coevolution framework (known as PGEC) and we address its 'inter-class' team behavior in comparison to that of CMGE. Chapters 6 and 7 provide the results of the comparisons outside of the EC paradigm. The CMGE framework is shown to

provide improved or competitive results under the majority of analyses, establishing the system as a viable classification framework for GP.

Chapters 8 and 9 analyze the behavior of the CMGE framework in terms of problem decomposition, establishing further distinctions from the 'ensemble' or 'weak learning' approach of PGEC. Moreover, the robustness of the system with respect to parameter (archive size) selection is investigated and some general recommendations are provided.

Chapter 10 provides discussion and summaries of the benchmarking results, highlighting the contributions of the CMGE framework toward addressing the fundamental pathologies of the GP algorithm under classification. We conclude with a set of recommendations for the use of the CMGE framework and provide directions for future work.

Chapter 2

Background and Related Work

This chapter develops background material pertinent to the thesis. As such, Section 2.1 reviews the particular form of Genetic Programming utilized in this work. Specifically, the method of Grammatical Evolution is assumed, although the contribution of this thesis is independent of the specific formulation of GP. The review of related literature begins in earnest in Section 2.2 where the scene is set for the supervised learning domain of classification under binary and multi-class contexts. In doing so, the case is made for multi-objective methods of fitness evaluation. That is to say, recent approaches to multi-class classification have begun to utilize multi-objective methods to 'co-evolve' each class simultaneously from a single population. In addition we question the extensive utility of Koza's 'switching' type wrapper function that is widely used in the classification domain as a mechanism for converting the original real valued GP representation into a class label. At this point we have made the case for: (a) pursuing a wrapper function based on a local membership function (Gaussian), thus opening up GP to the novelty detection model of classification as opposed to a discrimination based model of classification; and (b) utilizing a multi-objective methodology for facilitating the arbitrary decomposition of a problem into a set of class consistent mappings from subsets of the input exemplars to local membership function.

Section 2.3 returns to reviewing the broader literature in order to identify the most 'GP friendly' approach for performing Evolutionary Multi-objective Optimization (EMO). Such a perspective has been developed under a GA setting, resulting in various assumptions that do not necessarily carry over elegantly into the GP context. This section concludes by reviewing previous GP models that include an explicitly multi-objective fitness function. As the proposed local membership function approach

explicitly facilitates problem decomposition, we also provide a short review of previous approaches to automatic problem decomposition within the GP paradigm, Section 2.4.

At this point we have not explicitly dealt with scaling GP to larger data sets, although the review of appropriate EMO algorithms has enabled us to determine an effective scheme for defining early stopping. Section 2.5 provides a short survey of techniques that attempt to minimize the five factors identified in equation 1.2 from the introduction. From the perspective of this thesis, the work on competitive coevolution is the most relevant. The principal motivation of which is to evolve the learner population over some adaptive subset of the larger training set. The goal of such a methodology is to engage the subset population and learning population in an incremental game in which advances in classifier performance results in the subset of exemplars adapting such that classifier performance continues to improve.

The algorithm proposed in this thesis therefore uses a combination of: (i) competitive coevolution to scale the approach to larger data sets, while (ii) using a combination of co-operative coevolution (EMO) under a local membership function to guide evolution of GP individuals that decompose the problem into possibly multiple classifiers. All classifiers are evolved from a single population, without any *a priori* specification of the number of individuals to participate in a solution.

2.1 Grammatical Evolution

The following work is undertaken using a specific variant of GP, known as Grammatical Evolution (GE), however none of the algorithms are specific to the type of GP employed. GE was first introduced by Michael O'Neill, J. J. Collins and Conor Ryan in 1998 as an evolutionary algorithm that permits automatic and language independent evolution of programs of arbitrary complexity, provided that a grammar for the language can be supplied in Backus-Naur form (BNF) [91]. As opposed to the standard approach to GP, where programs are represented directly as tree structures, GE employs a linear genome in the form of a fixed length binary string which is used to derive a program according to an arbitrary, user-supplied BNF context free grammar (CFG). A high level representation of this process is illustrated in Figure 2.1. Figure

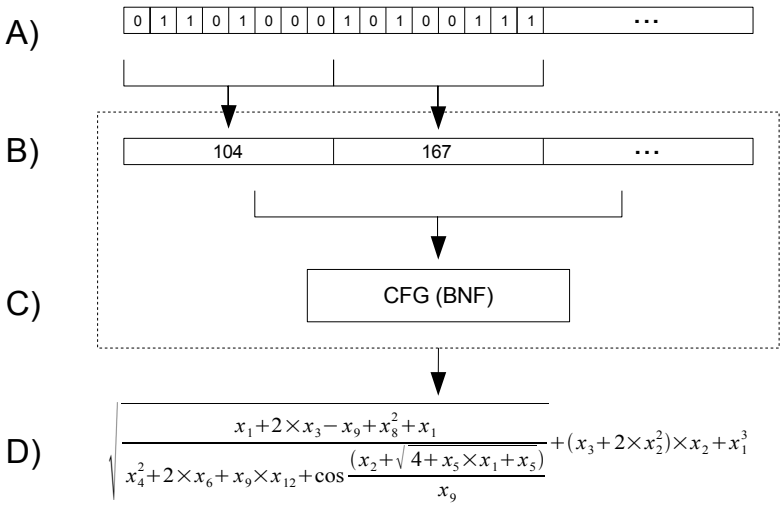


Figure 2.1: The GE decoding and mapping process.

2.1 step A) indicates the binary string being lowest level of GE representation. The string is of fixed length and should be evenly divisible into n-bit segments, or *codons*, where both parameters are chosen by the GE practitioner. The Figure 2.1 transition from A) to B) indicates the binary (DNA) to integer codon (protein) decoding process that is synonymous with the biological transcription process[1]. Steps B) and C) taken together form the GE mapping process, which is analogous to the translation of proteins to phenotypic traits or characteristics. The mapping process employs an internal, user supplied BNF grammar structure. The codons represent transitions or rule selections which deterministically expand non-terminal options to sentences in the language specified by the context free grammar; the pseudo code for this process is specified in Algorithm 2. Step D) represents the final expression or phenotype resulting from the mapping process.

While the fundamental motivation and end product of GE and traditional GP are largely the same, a key distinction involves the underlying program representation, specifically with regards to GE's use of an indirect mapping between the genotype

[1]Such a process is largely redundant within the context of the canonical GE model, with no variation in the alignment process to effect different decodings from the same codon sequence. As such, GE models generally begin with a sequence of integers with allele ranges spanning the set of predefined terminals / non-terminals of the CFG, as will be discussed shortly

(step B) and phenotype (step D). Unlike traditional GP, GE does not apply variation operators to the expression phenotypes themselves; rather GE programs are stored as a series of BNF grammar rule selectors, or *codons*, which are in turn represented by a fixed length binary string (i.e., a genotype). In this sense, GE is similar to Genetic Algorithms (GA) and consequently permits the use of simple GA search operators. This indirect mapping from genotype to codons to phenotype is analogous to the expression of genes in natural biology to proteins, which (in conjunction with other proteins) affect physical traits.

There are a number of critical implications of the genotype to phenotype mapping used in GE (the specific process of which is detailed in Algorithm 2):

1. The genotype-phenotype mapping is many to one, implying that changes at the genotypic level (due to application of GE's variation operators) need not affect the phenotypic representation (mapped program). In the case of the mutation operator this has been described as 'silent mutation' and is due in principal to the reliance on the MOD operator in the mapping of production rules. The application of the MOD operator ensures codons that take on values beyond the number of options for a production rule are mapped back to the legal range. This feature assumes sufficient redundancy in the codon representation; that is, codons must be defined to have the capacity to take on values greater than the number of options for rules under consideration. Greater redundancy in the codon representation can therefore increase the number of 'silent' mutations.

2. While a particular gene will always translate to a particular protein (codon), the protein may be expressed differently at the physical trait level (phenotype) depending on the presence or absence of other proteins (codons) appearing before or after it in the sequence;

3. By definition, the mapping process ensures that any terminating selection of rules as supplied by codon translation of the genotype will map to a syntactically correct program in the language (i.e., the phenotype). This property of the mapping algorithm provides an elegant solution to the closure requirement of traditional GP, where careful system design must be observed in order to

ensure that the outcome of any legal application of genetic operators results in a program that is syntactically valid;

4. A particular individual will always generate the same expression in the mapping from genotype to phenotype. Moreover, the mapping guarantees that the standard GA search operators can be employed independent of the problem.

Whereas the algorithms presented in this work are not specific to GE and because of the numerous similarities, the terms GE and GP will be hereafter employed interchangeably.

Algorithm 2 geMap (C) - Canonical GE individual mapping function.

Input: Codon array C. Employs local variables for option array (opt), next symbol (ns), codon pointer (cp) and stack S. Assumes global access to context free grammar G (e.g. Table 2.1). Local functions: isTerminal, option and #options are direct look ups given G; push, pop and isEmpty have the traditional implications for a stack.

Output: Expression string, e.

```
1:  cp := 0
2:  ns := <start>
3:  push( S, ns )
4:  while ! isEmpty(S) do
5:     ns := pop( S )
6:     if isTerminal( G, ns ) then
7:        e := strcat( e, ns )
8:     else
9:        opt := option(G, ns, C[cp] mod #options(G, ns))
10:       for i := |opt|-1 ... 0 do
11:          push( S, opt[i] )
12:       end for
13:       cp++
14:    end if
15: end while
```

For the classification algorithm presented here, individuals represent simple arithmetic expressions within the GE framework. The BNF grammar is defined by the 4-tuple $\{N, T, P, S\}$, where N = {<start>, <exp>, <preop>, <op>, <var>} and T = {(,), SIN, COS, SQRT, LOG, EXP, +, -, $\times$, $\div$}, sets of non-terminal and terminal rule symbols respectively; S = {<start>} is the (non-terminal) start symbol; P

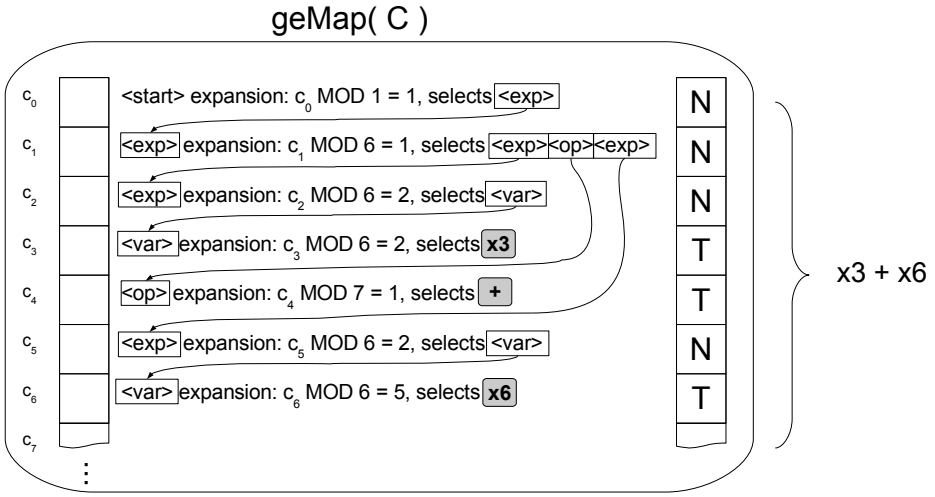


Figure 2.2: Example mapping codon-level representation to phenotype expression.

is a set of production rule transitions, mapping N to T. A set of production rules for the classification grammar for a problem having n features (or attributes) is provided in Table 2.1. A sample GE mapping from a codon string $C_0 \ldots C_6$ to the phenotypic expression x3 + x6 according to the grammar supplied in Table 2.1 is illustrated in Figure 2.2.

2.1.1 Genetic Operators

Implicit in the linear mapping characteristic of GE is the high degree of linear genetic dependence and therefore the potential for widely disruptive impact of each gene modification. That is to say, the genotype is incrementally converted into the corresponding phenotype using the CFG to convert non-terminals into terminals, where non-terminals frequently expand recursively to additional non-terminal symbols before the grammar identifies a terminal[2]. For example, a single bit mutation when applied to the binary representation of a non-terminal-expanding codon, can dramatically impact the phenotypic expansion for all of the following genes due to the change of context. Similarly, the traditional crossover operator will result in code exchange

[2]The codon expansion type information (e.g., terminal versus non-terminal or non-terminal type) is readily accessible in the mapping process (lines 7 and 9 of Algorithm 2); this is illustrated by the 'N' or 'T' following each expansion step in Figure 2.2.

between individuals without regard to context information. This has led to the criticism that GE variation operators impose much more destructive changes than their GP counterparts. The result has been the recent development of context sensitive crossover operators that note the gene expansion types (where types correspond to non-terminal expansions vs. terminals) during the mapping of the phenotype [47].

Context sensitive variation operators are explicitly defined for crossover and mutation such that the traditional operators of single point crossover and bit-wise mutation are constrained in their application based on codon expansion types. Matching Crossover [47] restricts crossover to codon boundaries where the codon of the second parent expands the same non terminal type as the codon selected in the first parent. The expectation of context preservation is therefore based on each parent receiving material that is used to interpret the same part of the grammar as the donating parent. While this does not preclude disruptions later in the sequence, it does provide a basic means to guarantee preservation of context at the crossover boundary and is readily applicable to the wrapping form of GE. Such a crossover operator was demonstrated to perform significantly better than the single point crossover typically employed by GE [47].

Under a similar use of the expansion type information, mutation can be readily applied to avoid structural disruption by restricting application to genes corresponding to bits of terminal-mapping codons in the phenotype. This variation on the standard mutation directly results in alternative terminals of the same arity as opposed to potentially substantial structural changes in the phenotypic expression.

2.2 Classification

Pattern classification is a central task in machine learning that involves prediction of a class label based on a set of numeric and/or nominal-valued inputs (features or attributes) associated with each exemplar. Classifier induction under the standard GP framework is a supervised machine learning task that employs a set of pre-classified exemplars (i.e., a labeled training set) with the goal of producing a simple classification rule which can readily generalize to unseen instances post-training.

In the literature, a distinction is typically made between binary classifiers and

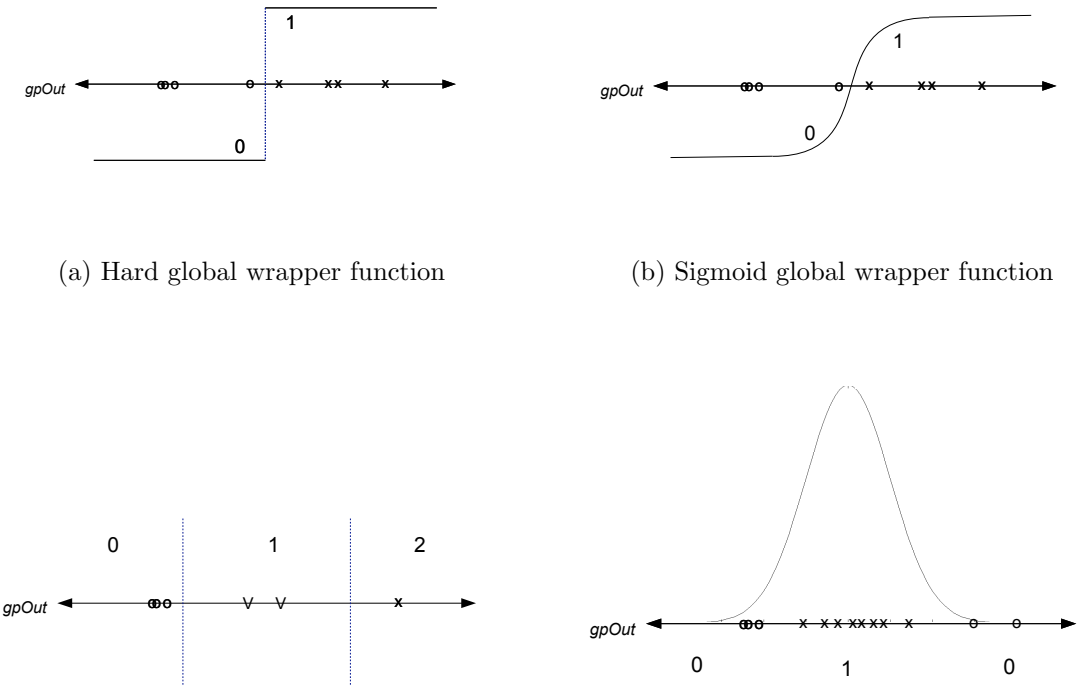
Table 2.1: Sample context free grammar for classification under GE

Rule ID	**Rule**	**Option**	**Option ID**	**# Options**
0	<start>	<exp>	0	1
1	<exp>	<exp>	0	5
		| (<exp>)	1	
		| <exp> <op> <exp>	2	
		| <preop> <exp>	3	
		| <var>	4	
2	<preop>	SIN	0	5
		| COS	1	
		| SQRT	2	
		| LOG	3	
		| EXP	4	
3	<op>	+	0	4
		| -	1	
		| $\times$	2	
		| $\div$	3	
4	<var>	x_0	0	n
		| $\vdots$	$\vdots$	
		| x_{n-1}	$n-1$	

multi-class (or multi-category) classifiers. A multi-class classifier predicts one (or multiple) labels for each example from a set of many candidate labels, while a binary classifier predicts each example as either in or out-of-class. While the classification task has received considerable attention in the GP community, several problems remain universally acknowledged, specifically with respect to how GP may be readily employed in large, 'real-world' classification problems. In such instances, training sets consist of tens or hundreds of thousands of exemplars and the class distributions may be widely unbalanced. Ensuing solutions must provide good generality (performance on unseen exemplars) yet remain computationally tractable. Moreover, many real-world classification problems are multi-class (as opposed to binary) in nature and potentially require several class-specific expressions to collaborate in order to provide adequate coverage over all classes. The conventional approaches to classification within the GP paradigm do not adequately address these questions in general, and the current work is intended to provide a more comprehensive framework for classification under the GP context. In the current work we employ tools from the evolutionary multi-objective and coevolution literature to achieve scalable, multi-class solutions that decompose the problem from a single population.

2.2.1 Binary Approaches

Binary classification problems under the GP paradigm are typically configured to calculate a single value that is used to characterize the quality of an individual in terms of a specific performance metric (e.g., accuracy, sum squared error, weighted combination of sensitivity / specificity etc. . .) [75] [12]. This value is then scaled, providing a fitness value that indicates the individual's probability for selection during evolution. This naturally leads the search to a single individual having the largest fitness value, which is then used to solve the classification problem alone. The implicit assumption in this model however, is that one expression (moreover, a single objective) is sufficient and/or appropriate to solve the classification problem. Moreover in the case of binary problems, classification decisions under GP conventionally take the form of a hard switching function arbitrarily centered at zero. GP outputs greater



(a) Hard global wrapper function

(b) Sigmoid global wrapper function

(c) Range-based global wrapper function

(d) Gaussian local membership function

Figure 2.3: GP classifier wrapper functions: (a) Hard global wrapper function, (b) Sigmoid global wrapper function, (c) Range-based global wrapper function, (d) Gaussian local membership function (LMF).

than zero indicate 'in-class' patterns, while those less than or equal to zero are considered to be 'out-of-class' (Figure 2.3 (c)). George et al. [43] examined alternative wrapper-fitness functions (including sigmoid / SSE and a wrapper-less / separation distance) in the binary context and concluded that a wrapper-less approach provided the most dependable results in terms of accuracy and variation. In the wrapper-less formulation only the class separation distance was considered in fitness assignment, which was defined by mapping the raw GP output on each class and calculating inter-class separation distance. Post-training, labels were assigned according to the nearest-neighbor based on distance to in / out-of-class cluster means.

Such a methodology explicitly questions the utility of the wrapper operator.

Specifically the GP wrapper operator typically employed takes the form of a switching function (Figure 2.3 (a)). As indicated above, such an operator 'quantizes' the original mapping performed by GP from a typically multi-dimensional input space to a one-dimensional output space. The quantization corresponds to the number of classes, where Koza only considered the binary case, whereas [75] considers the case for multi-class quantization (Figure 2.3 (c)). Such an operator results in a fitness (cost) function that counts the number of correctly classified exemplars, but does not distinguish between an individual that finds a mapping that explicitly pushes points away from the switching threshold, and one that leaves points near this threshold. This is significant because performance on unseen exemplars might well be worse in the case of the latter, but result in more 'robust' or generalized behavior in the former. The work of George et al. is interesting in this respect because it effectively dropped the concept of a wrapper operator entirely, thus also avoiding the requirement to *a priori* specify the point at which 'class switching' appears.

More generally, by considering the classification problem as finding a series of mappings from the input space to a one dimensional output space, we are able to rephrase the problem as locating a set of mutually exclusive clusters that decompose the original set of class consistent (input) exemplars when mapped to the one dimensional output space. The ensuing clusters on the GP output space imply that possibly multiple individuals cooperate to find the required decomposition of the classification problem i.e.,, each individual searches for a class consistent cluster that minimizes the overlap with other individuals, while maximizing (minimizing) the number of in-class (out-class) exemplars described per cluster. Finally, we also recognize that such a framework implies that each cluster be described in terms of a local membership function, such as a Gaussian, thus the resulting classification model represents a novelty detector as opposed to a discriminator. The latter point is significant as most machine learning algorithms assume a discriminator based model for establishing decision boundaries. Such a scheme however, may result in unpredictable behaviors on unseen data [79]. In particular discriminator models rely on 'global activation' functions such as sigmoidal operators (Figure 2.3, (b)) or the switching type of wrapper of Koza's canonical GP classifier (Figure 2.3, (a)). The unfortunate consequence of this

is that when such a model encounters unseen exemplars that do not belong to the distribution of data encountered during training (e.g., as in a fault condition or outlier), the discriminant based classifier is forced to label the exemplar as in or out-of-class, whereas what is more desirable would be for the model to declare such exemplars as neither class [79]. In effect by assuming the proposed cluster decomposition approach to classification we are able to also provide a more informative model for classification than has previously been possible under the GP paradigm. Such a novelty detection based framework may be particularly appropriate for domains in which it is difficult to provide representative samples of training data for all out-of-class conditions, for instance, medical diagnosis, intrusion detection, and fault detection.

2.2.2 Multi-class Approaches

The standard binary approach to classification has previously been extended to multi-class problems by combining binary classifiers to form teams or hierarchies with one or more expressions assigned to each class of the problem under consideration [58]. This normally requires a cumbersome set of separate initializations for each class, although several single-population approaches have surfaced recently [75] [13] [12] [105] [80].

The standard approach to multi-class is known as Binary Decomposition, where the approach evolves separate binary classifiers for each class [58]. This necessarily implies separate initializations of GP for each class in the problem under consideration; dividing this job across a Beowulf cluster, for example, provides a brute force method to arrive at a classifier for an arbitrary number of classes on the same data set.

A more direct approach to the multi-class problem was taken under a constrained syntax representation of GP [13] [12], where multiple models (one per class) are the result of a single initialization of GP. The representation imposed syntactic constraints to evolve simple conditional rules that could be applied to any pattern. Individuals were then assigned a class deterministically as that which rewarded the individual with the best composite fitness (a composite function was employed to encourage solution transparency in addition to the usual accuracy objective). Class-wise elitism

was enforced to avoid the occurrence of convergence on a single class and the winning set of individuals are combined to produce a final solution. Under the post-training combination scheme, any pattern that satisfies more than one rule is assigned to the class of the rule having the best training accuracy and those satisfying none are assigned to the majority class. A sensitivity / specificity analysis on a very small data set was presented indicating preferential results in comparison to the C4.5 decision tree.

Smart and Zhang propose the Communal Binary Decomposition (CBD) approach [105] that builds on the Binary Decomposition of Kishore and the Probabilistic Multi-class (PM) model of Langdon and Poli [67] for GP classification. The PM approach is based on probabilistic models of GP program outputs. Specifically, a GP is run over each class of the training data to determine the mean and standard deviation of each class on the output range. In a binary problem, the class separation distance is employed to arrive at a fitness value directly; in the case of a multi-class problem, fitness is based on a sum of pairwise distances between all combinations of class distributions. This method still constrains each program to solve the entire problem. In contrast, CBD assigns fitness to an individual based on the single class resulting in the best binary class separation distance and maintains an elitist archive of the single best result (i.e., expert) for each class. When a class expert provides separation distance beyond a pre-selected threshold, the class is no longer considered in the fitness assignment thus providing a means of class-wise early stopping criteria. CBD cassifier voting is a multi-class approach (one expert per class) from a single run of GP; combining of classifiers is therefore achieved using a (Gaussian) probability density function related to each expression. CBD overall accuracy results compared favorably against PM and a static range selection approach (discussed below) over four image classification problems, however training time was longer (typically many times longer) for CBD than the alternatives.

Loveard and Cieisielski described four approaches which provide single expression solutions to multi-class problems in a single initialization of GP; two of the approaches require specialized formulations of GP (Class Enumeration and Evidence

Accumulation) and two are range selection techniques (static and dynamic range selection, or SRS and DRS respectively). In the SRS approach, a series of intervals over the GP output range are chosen *a priori* to be associated with each class. The DRS version chooses intervals dynamically based on output from the GP programs themselves given a subset of the training data. Class enumeration involves a new conditional structure and further requires all trees to return terminal values corresponding to a class label. Evidence accumulation employs a secondary data structure for each tree, known as a *certainty vector*, which contains a numeric accumulator element for each class. A novel form of terminal which adds or subtracts a constant from the elements of the certainty vector during program execution (per exemplar) is introduced. Following program execution the certainty vector is examined for the largest element, which indicates label assignment. The four approaches were evaluated against the standard binary decomposition approach to multi-class classification [75]. In both the binary and multi-class context, the DRS approach was reported to have best accuracy given comparable training times. The binary decomposition method was typically as accurate, however will generally require considerably longer training times as the number of classes increase. Moreover, the SRS method has been successfully applied to real-world, multi-class image classification problems including coin recognition and retinal pathology [120].

Muni et al. introduced a 'multi-tree' approach to multi-class classification using GP whereby individuals take the form of multi-trees, i.e., one tree is provided for every class under consideration within each chromosome [87]. The classification is made by examining the root values at each tree and assigning the class label based on the output having a positive value (essentially applying the standard hard wrapper function to all raw outputs). In the event of contradictory output (i.e., multiple trees (or none) evaluating to positive outputs), a conflict resolution heuristic was employed to arrive at a label. In this sense, the multi-tree approach provides a single solution (composed of multiple models – one per class) from a single run of GP. Results across five benchmark data sets in terms of overall classification accuracy were competitive with other forms of GP as well as various tree, Neural Network and statistical classifiers, however the approach requires a specialized representation

(and associated operators) along with many non-standard parameters, heuristics and optimizations; moreover there were no specific provisions for training on large or unbalanced data sets.

2.3 Evolutionary Multi-objective Optimization

Evolutionary multi-objective optimization (EMO) represents a relatively recent approach to solving problems having more than a single objective using the methods of evolutionary computation. The first practical EC approach that addressed problems of the multi-objective nature directly was Schaffer's Vector Evaluated Genetic Algorithm (VEGA) [103] [104]. In principle VEGA did not modify the standard GA approach aside from the selection mechanism. In turn, each objective would be used to proportionately select individuals and create sub populations which were later merged to compose the population for the following generation.

Aside from VEGA, multi-objective search and optimization problems were historically configured in one of two ways under the evolutionary computation context [26]. In the first approach a single, composite objective is employed as a weighted combination of the original objectives. The second approach chooses one of the objectives and formulates the remainder as constraints having pre-defined limits. The usual evolutionary procedures may then be applied directly. The principal drawback being that the weights or limits must be chosen *a priori* and may affect the quality of the final solutions.

In the following, we briefly survey some of the milestones in the development of Evolutionary Multi-objective Optimization (EMO) methods as they developed in the GA context; we will then revisit the problem from the perspective of GP, where EMO has only recently begun to borrow from the GA context. From the GP context we will use EMO to provide the basis for problem decomposition as well as MO problem formulations and therefore move on to survey other methods for encouraging problem decomposition in Section 2.4.

2.3.1 Pareto Dominance

The notion of Pareto optimality under the EC context was introduced by David Goldberg [44] as a method for considering each of the multiple, possibly conflicting, objectives simultaneously in order to obtain a set of mutually equivalent candidate solutions (i.e., the Pareto optimal set) that trade off in their optimality across objectives. In the absence of information defining specific preferences for the objectives, the Pareto optimal set contains the set of alternative, mutually optimal solutions (those trading in one objective for another) which must be chosen from by a *decision maker* (DM) [37] based on some final selection criteria.

The Pareto approach defines a partial ordering of candidate solutions based on the notion of Pareto dominance. An individual A Pareto dominates individual B over the set of objectives Θ if A is no worse than B on any objective ($\theta \in \Theta$) and is strictly better than B on at least one [66]. Formally for the case of minimization (i.e., $A \prec B$):

$$A \prec B \Leftrightarrow \forall\theta \in \Theta \quad A_\theta \leq B_\theta \wedge \exists\theta \in \Theta : A_\theta < B_\theta \tag{2.1}$$

A and B are said to be incomparable when A does not Pareto dominate B and vice versa and an individual is non-dominated when it is not Pareto dominated by any others[3]. The set of (incomparable) non-dominated individuals constitute the *Pareto Set*.

2.3.2 ϵ Approximations to Pareto Dominance

The definition of Pareto Dominance can be generalized using the notion of ϵ-*dominance*, which effectively provides a general means to discretize the objective space and constrain the number of elements in the solution set to be finite. The significance of this lies in the guarantee of diversity among solution vectors and practical size of the solution set, known as the ϵ-approximate Pareto set [68]. Multiplicative ϵ-dominance (for minimization $A \prec_\epsilon B$) is defined as:

$$A \prec_\epsilon B \Leftrightarrow \forall\theta \in \Theta \quad A_\theta \cdot (1-\epsilon) \leq B_\theta \quad | \quad 0 \leq \epsilon < 1 \tag{2.2}$$

[3]When A and B contain identical values in all objectives they are not considered incomparable; rather these cases are termed *indifferent* and are handled separately in the current work.

Moreover, an additive variant can be employed when specification of constant ϵ values is desired across objective dimensions:

$$A \prec_\epsilon B \Leftrightarrow \forall \theta \in \Theta \quad A_\theta - \epsilon_\theta \leq B_\theta \tag{2.3}$$

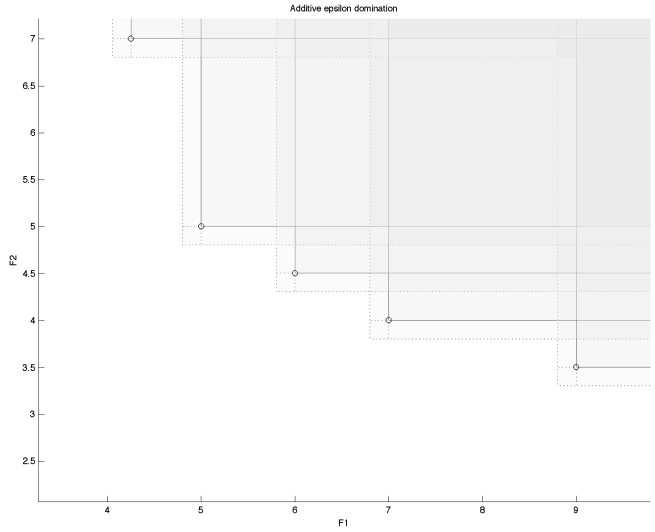
The additive ϵ-dominance definition provides uniform discrete resolution over the objective space as opposed to the multiplicative variant, where larger hyper-rectangular cells (corresponding to lower objective space resolution) are defined as dimension magnitudes increase. This contrast is illustrated in Figure 2.4.

2.3.3 Pareto Ranking, Diversity and Elitism

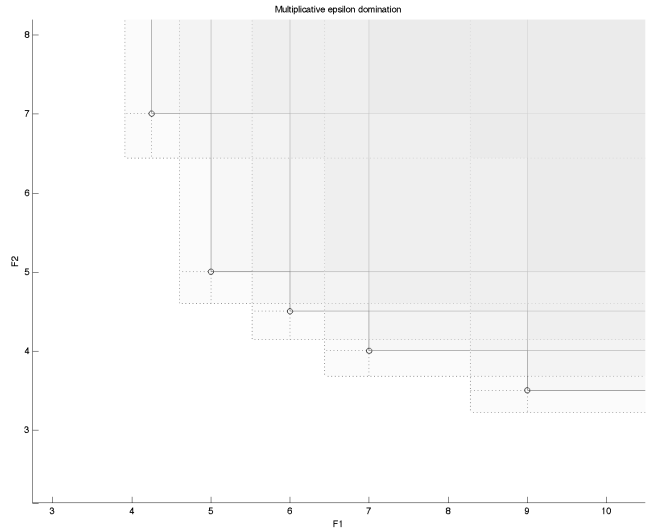
The notion of Pareto optimality, in terms of ranking and preservation of diversity as introduced by Goldberg [44], has inspired several generations of EMO algorithms [20]. Recent advances in the literature have demonstrated that a Pareto front may be used to maintain a set of candidate solutions to multi-modal problems [26]. Within the GA context, candidate solutions describe a point in multi-dimensional space, where the basic objective is to locate the set of solutions (points) minimizing a predefined objective over an unknown (multi-modal) function. Three fundamental goals are common to the design of modern EMO algorithms, specifically [123]:

1. Ensure that evolution guides solutions toward Pareto optimality;
2. Maintain diversity among solutions in the objective space;
3. Prevent the loss of non-dominated solutions.

A Pareto ranking mechanism, based on Pareto dominance as described in Section 2.3.1, provides the basis for the ensuing fitness assignment scheme and guides the evolutionary algorithm by favoring non-dominated solutions in the selection and reproduction processes. Typically the mapping from rank to fitness is linear or exponential but varies by algorithm. Moreover, in order to avoid dense regions in the objective space, a diversity preservation mechanism may be included where this typically takes the form of preselection, crowding, or sharing (where the latter requires additional parameterization in the form of a sharing radius) in the genotype or phenotype space.



(a) Additive ϵ-dominance



(b) Multiplicative ϵ-dominance

Figure 2.4: ϵ-domination regions for the case of a) additive and b) multiplicative variants.

When these are combined with an explicit archiving structure (having acceptance criteria based on non-dominance) or an elitist replacement scheme, a diverse collection of non-dominated solutions are readily produced by the EMO algorithm. Among the most significant and representative of the recent of the EMO algorithms are:

1. Strength Pareto Evolutionary Algorithm (SPEA / SPEA2) [125] [126] [124];
2. Nondominated Sorting Genetic Algorithm (NSGA [112] / NSGA-II [27]);
3. Multi-objective Genetic Algorithm (MOGA) [37];
4. Pareto Converging Genetic Algorithm (PCGA) [66].

These algorithms have been employed extensively in the literature and all achieve similar goals as outlined above; however, the algorithms vary with respect to the approach taken to their ranking mechanisms, maintenance of diversity and prevention of solution loss. The four algorithms indicated above will be discussed in turn with specific focus on these distinguishing features.

SPEA / SPEA2

The original SPEA algorithm was introduced by Zitzler et al. and is a generational algorithm that employs a solution archive which acts as a fixed-sized external memory preventing the loss of non-dominated individuals. At the start of each generation the non-dominated members are updated to the archive, culling dominated and indifferent members accordingly. In the event of an archive overflow, additional members are pruned on the basis of a clustering algorithm which preserves diversity over the non-dominated front. During evolution, archive members are assigned a strength that is proportional to the number of population members that they dominate over the target objectives, while population members receive a fitness according to the strengths of archive members by which they are dominated. In this way, only a weak density relationship is enforced among population members. Moreover, this implies that the fitness assignment and diversity maintenance is highly dependent on the archive size and contents. These issues were addressed in SPEA2 by providing a more fine-grained fitness assignment and incorporating an explicit measure of density information (by

way of k-Nearest Neighbor clustering) which directly influences selection over the archive members.

From the perspective of using SPEA as the basis for GP EMO, we are immediately confronted with the problem of defining an appropriate process for diversity maintenance; SPEA assumes the availability of a clustering algorithm. In GA this is not a problem as the genotype denotes a point in the domain feature space; comparing the 'location' of two individuals with regards to establishing cluster membership is therefore trivial. Conversely, genotypic similarity under the GP model is not straightforward. As such, one is often reduced to comparing the behavioral properties at the phenotypic level, a costly activity as it implies comparison over the training data.

NSGA / NSGA-II

The NSGA family of generational algorithms introduced by Srinivas and Deb implement a non-dominated ranking scheme as a means of influencing fitness assignment [112] [27]. In the original variant of NSGA, the non-dominated ranking procedure finds all non-dominated solutions and assigns them to the highest rank. Fitness is then assigned within the rank as a genotypic share of the highest 'dummy' fitness value (thus encouraging diversity among members of the rank). These individuals are removed from consideration and the process is iterated. NSGA-II improved on the original by introducing improved ranking and diversity maintenance schemes. Preservation of non-dominated individuals was due to an elitist inter-generational maintenance of best individuals. That is to say, succeeding generations are populated by a *crowded-comparison* operator among the union of parents and children which prefers high ranking solutions but employs a crowding distance to favor solutions from sparse regions of the objective space for members of the lowest rank [27].

MOGA

Fonseca and Fleming's MOGA suggests a subtly different ranking scheme from that of Srinivas and Deb in which all members of the population are assigned a rank that is equal to one plus the number of individuals by which they are dominated *without*

removal after each rank. Members having rank 1 (that is, the highest rank) are assigned the highest fitness while members of following ranks are assigned fitness as a linear function of rank, scaled by objective space density in the form of niche counts according to a sharing distance [37] [20].

PCGA

Of specific interest to the GP EMO proposed in this work is the steady-state PCGA algorithm introduced by Kumar and Rockett [66]. PCGA assigns ranks in a similar fashion to MOGA, assigning fitness as a linear function of rank. However the current approach deals with ties in the objectives between two individuals by increasing the rank of one, chosen at random. This method is described as *ranking with ties* [66] and forms the basis for fitness assignment used in the present work. Moreover, as a steady state algorithm, PCGA enforces elitism at each incremental change in the population, replacing the lowest ranking member only if it is lower ranking than the offspring. As such the authors do not employ a diversity mechanism explicitly; arguing that the compressed range of ranks due to the elitist steady-state strategy of PCGA produces the same effects as fitness sharing [66]. In further motivation to the current work is PCGA's use of early stop criteria in the form of difference of *rank histograms*. Pareto rank histograms are defined as a frequency distribution of tied ranks in the population and are generated for successive epochs from the ratio of number of individuals at a given rank in the current population to those at the same rank in the combined and re-ranked populations of the current and previous epochs; a match between successive rank histograms indicates an appropriate point for early stopping [66].

In summary, from the perspective of a GP EMO model, the PCGA algorithm side steps the entire issue of niching via distance or similarity metrics. Instead this property is supported through an elitist steady state mechanism applied to the Pareto ranking process. This approach is much more applicable to the GP context in which establishing similarity cannot be performed genotypically, but requires a costly phenotypic comparison of the behavior across training data.

2.3.4 Multi-Objective Genetic Programming

Multi-objective optimization techniques have recently been applied to the GP context toward parsimony and diversity enforcement by framing these as explicit, independant objectives. Encouraging results on the Even Parity test problem [61] have been published in [11] where the SPEA2 algorithm is employed using a parsimony objective (minimization of solution size) in addition to the standard fitness objective. De Jong, Watson and Pollack's Find Only and Complete Undominated Sets (FOCUS) algorithm [25] additionally enforced a diversity objective (maximization of diversity) on the even 3, 4, and 5-Parity problems with results that outperformed standard GP in terms of computational effort and solution size; however, FOCUS also employed a special phenotypic distance metric that involves summing the distances between nodes of corresponding trees (each node difference increments the distance by 1 when corresponding trees are overlaid starting from the root) [25].

One of the first attempts to apply MO techniques to a 'real-world' GP problem domain was perhaps [99] where the MOGA algorithm was directly implemented in the GP context, however this was specifically aimed at the regression task. Additional examples of MOGP have surfaced recently in the regression domain (e.g., the Ordinal Pareto GP of [107] using error and complexity objectives) however we will focus the remaining discussion of MO techniques as applied to the GP context on the classification task.

Limited instances relating MO methods of GP to the classification task have appeared in the literature. Pseudo-Objective Parsimony Enforcement GP (POPE-GP) and Decomposed Multi-Objective GP (DecMO-GP) have been applied to classification tasks involving small and near ideally balanced data sets including UCI Breast, Iris and Wine [93]. Both algorithms employ the NSGA II algorithm in the search with POPE-GP using an explicit parsimony objective under the classification task while DecMO-GP decomposes the standard error minimization objective into separate error minimization objectives for each class under consideration. DecMO-GP is extended to include a parsimony objective in the DecMOP-GP algorithm. Lower error rates and computational effort in comparison to the standard GP approach were reported

over all three algorithms on the classification problems. In all cases the SRS framework (Section 2.2.2) employed provides single, 'super' individual solutions though the authors acknowledge a largely heuristic approach to the final solution choice and indicate interest in the multi-model, hierarchical approach proposed in [80].

Zhang and Rockett have recently employed the MOGP context to evolve 'optimal' feature extractors to drive a separate classification stage [121]. The process involves transforming the input space into a one dimensional decision space such that pattern separability in decision space is maximized. Under both generational and steady-state contexts, they employ a fitness function that was based on Pareto comparison of three variables (complexity, bayes error and misclassification error). Ensuing classification results on five UCI problems appeared to be competitive with results reported in the recent literature, however between the two approaches investigated the errors returned were not significantly different. It was reported, however, that the steady-state approach (referred to as PCGP), returned much smaller solutions. The classifiers of this work still appeared to result in hard thresholding and single, 'super' individual solutions that were applied to the binary context only.

Of specific interest in this work is to provide a methodology for utilizing the MO framework, currently demonstrated under a GA context, for solving the problem decomposition issue under Genetic Programming. To do so we make use of a recent result from GP in which the switching (or global) wrapper classically employed to map the 'raw' GP output to a discrete number of class labels is replaced with a local or Gaussian-type wrapper function [43]. Under this context, it is now possible to phrase the classification problem as finding the minimum set of mappings from a multi-dimensional input space to class consistent clusters on the one-dimensional GP output space.

Such an approach enables us to leave the problem of finding appropriate mappings and matching exemplars to mappings to EC. Moreover, desirable properties for the (and therefore mappings) are described in terms of the PCGA framework enabling us to make use of the associated Pareto based early stopping criterion. However, unlike Zhang and Rockett we do not limit each class to a single mapping; in effect we have a process for automating problem decomposition. In the following section we

summarize work to date directed at problem decomposition under the GP paradigm.

2.4 Problem Decomposition

Since the conception of Genetic Programming several methodologies have been proposed for encouraging solutions to take the form of a set of programs (individuals) solving different parts of the problem, as opposed to the population converging on a single 'super' individual. Recent examples might include the cooperative [95], and competitive [24] co-evolutionary paradigms, where both have been demonstrated within a Genetic Algorithm (GA) context. Both co-evolutionary approaches are multi-population models. In this work we are specifically interested in developing a multi-member solution from a single population, as opposed to 'super' individuals, where the latter has recently been investigated in a Multi-objective GP context for classification [93] [121] and regression [99] [107], as indicated above.

2.4.1 Automatically Defined Functions

Under the tree-based GP context, Koza enabled a form of intra-individual problem decomposition through the use of Automatically Defined Functions (ADFs) [62]. ADFs permit a form of problem decomposition within each individual through the explicit specification of a tree-representation that incorporates modular sub components or functions. ADFs require a main, result-producing branch, along with a user-defined number of branches to be co-evolved as functions having an assumed number of arguments; as such, the ADF specifications are highly representation and problem dependent. Moreover, the degree of problem decomposition achieved within an individual is directly influenced by the ADF architecture specification rather than an emergent property of the GP itself. Finally, when applied to problems that did not need the 'complexity' introduced by ADFs, the canonical GP model was found to perform better [62]. In effect, a user would have to run the canonical form of GP with and without ADFs in order to understand whether they were necessarily helpful in addition to experimenting with the additional parameters of argument counts and function sets.

2.4.2 Decomposition by Teams

Decomposition by teams refers to an inter-individual, multi-model problem decomposition approach based on the evolution of teams of competing or cooperating models that enable a degree of partitioning of the solution space. Moreover, teams can be homogeneous (e.g., N copies of the best individual) or heterogeneous (N different programs contributing to a final solution). The latter is of relevance to the current thesis, where models (i.e., team members) may be specialized within the problem domain or provide a degree of reinforcement (overlap) in their behavior, or both as necessary to perform better than a single individual.

The evolution of teams has been a subject of numerous studies in the literature and can be implemented in a variety of ways. In general there are three fundamental design decisions involved in team approaches: how to select team members, how to combine solutions, and how to assign credit [16]. Banzhaf's taxonomy adequately characterizes the main approaches as follows:

1. Random team member selection from the same population [109]. While this provides a straightforward heuristic to member selection an alternative, greedy selection scheme has also been employed in the ensemble context [41]. The primary stumbling block to member selection approaches is associated with the credit assignment problem (see Section 1.4.5), where an appropriate fitness assignment algorithm is required to distribute credit according to contribution of individual participants (i.e., potential team-members). That is to say, how can fitness be efficiently assigned so as to properly reward (and thus pursue) team members contributing to strong performance?

2. Team members evolved in separate sub populations (also known as the island approach [111]) e.g., non-dominated random sub-population component selection of [53]. In practice the combination of team members from several populations or niches raises a number of additional problems, including team selection and output composition; although the latter has been successfully addressed, for example, by the bidding models of Lichodzijewski et al. in [73]. In other words, the choice of individuals to include in a solution can dramatically affect

system performance, as can the choice of how to coordinate their collaborations [16]. The island approach can produce reasonable teams having members with a high degree of fitness and independent errors; however, this requires that all solutions are equivalent (no solution bias) to avoid convergence on the same (easy to find) solutions [111].

3. Individuals as explicit team representations within the evolutionary process (also known as the team approach [16]). The main drawbacks of this approach include the requirement of appropriate team representation (plus relevant team operators) and proper assumptions on team size. Soule reported that teams evolved under this configuration can exhibit good overall performance using small team sizes, with inversely correlated errors and a high degree of specialization (or cooperation) but may suffer from poor fitness among individual members. As a result, overall solution performance was shown to deteriorate with increasing team sizes [111].

4. Island / team hybrid approach. Soule recently presented an interesting variant known as the Orthogonal Evolution of Teams (OET) which simultaneously applies evolutionary pressure for team performance and member performance [111]. The OET approach employs an island model (one island per team): first a selection operator chooses highly fit team members (per island) as parents to produce new teams of pre-defined size. The update scheme inserts the new (offspring) teams into the associated island, replacing teams having of low fitness. The implicit assumption appears to be that the credit assignment problem can be appropriately solved (to accurately reward individuals with respect to their team performance) using their isolated performance. That is, the approach assumes that team members with high fitness when tested individually will lead to good (uncorrelated) team members without explicitly making this a requirement for member selection. This idea conflicts with results presented by Soule in [110], where team member fitness was typically poor. In other words, the most fit individuals may not lead to the best teams.

In [94] a multi-model, majority voting scheme for GP (Majority Voting GP Classifier, or MVGPC) is investigated. Sets of voters (models) were collected as best-of-population individuals over multiple runs of GP; in the case of multi-class problems, the standard binary decomposition approach (separate initialization per class) is employed over multiple runs. An individual's vote increments the in or out-of-class counter for the class corresponding to the voter. Labels are assigned based on the class having the largest ratio of in vs. out-of-class counts. A simple heuristic is employed for conflict resolution (automatic assignment to the lower of conflicting classes) and a special 'misclassified' label is assigned when all ratio values are 0. MVGPC demonstrated superior performance on Leave One Out Cross Validation (LOOCV) experiments versus a k-Nearest Neighbour implementation on two real-world gene expression problems having small, unbalanced numbers of exemplars and thousands of features. The authors demonstrated that the best results were obtained when the number of voters was equal to the number of training patterns, however larger data sets (in terms of number of training exemplars or total number of classes) would directly preclude this approach based on the demanding computation (training) requirements.

The cooperative, team-based approach investigated problem decomposition from the perspective of several combining policies, notably including the coevolution of voting weights and Neural Network optimization to achieve specialization on binary classification and regression subtasks [16]. Here, Brameier and Banzhaf take the 'Team Representation' approach using the linear GP model where a specialized representation is devised for teams (i.e., a fixed number of programs defined per individual) and appropriate team recombination operators are introduced. Fitness is a composite function of overall team error and a weighted sum team component errors. A total of seven approaches to linearly combining the outputs of team members are presented, where 'output' is either based on the raw expression value or wrapped class decision, depending on the approach. Combining policies included constant weighting (or essentially averaging of outputs), normalized error (i.e., a function of fitness) as weights, cooperative coevolution of weights, majority voting (MV), winner-take-all

(WTA), and a linear perceptron optimized weights (applied to raw output). Significant improvements on classification (train and test error) over three problems by all the of team approaches over standard GP were reported, with only moderate increase in overall solution complexity.

In summary, other than the bidding models of Lichodzijewski and Heywood [73] [72], the above models for problem decomposition all assume that the number of cooperating models is pre-specified. It remains to be seen, however, whether the bid-based mechanism can incorporate properties such as multi-objective fitness functions, where this is frequently a benefit for establishing parsimonious solutions.

2.5 Scalability

The ability to train the GP algorithm on large data sets is frequently cited as a primary disadvantage of GP in comparison to other machine learning approaches. Scalability, particularly with respect to real-world and multi-class problems, remains a widely acknowledged issue in the classification context of the GP paradigm, where recent approaches in the literature have invoked:

1. Systematic, incremental reduction of population size during evolution [76];
2. Parallel and hardware-specific speedups [90] [7] [36];
3. Sub sampling of the training data [42] [108] [22] .

These approaches to scalability will be discussed and compared in turn over the following sections.

2.5.1 Variable Population Size

Luke et al. employ a population implosion strategy, where population size is decreased over the course of a GP run to effectively reduce run time while maintaining (or improving on) the results of the standard approach [76]. Specifically, the *layout* (describing population versus runtime) is explicitly defined to be diagonal as opposed to rectangular, such that the population size is reduced linearly over the run as opposed to remaining constant, as is the case in a traditional run of GP. The diagonal

layout was compared with several traditional rectangular layouts over four standard GP benchmark problems and in the majority of experiments the former showed significant performance improvement when run over pre-specified numbers of evaluations. The primary performance improvement reported is achieved through the reduction in population size, which explicitly addresses the evaluation function as specified in Equation 1.2 by lowering the number of population members needing evaluation, particularly late in the run.

A similar result was obtained by examining the effect of *plagues* on GP populations [33], where Fernandez et al. reported to reduce computational requirements while maintaining the quality of the resulting solutions employing a plague metaphor. Plague is used to refer to the process of systematic elimination of individuals (those having the lowest fitness) along generations. While effecting performance gains in line with improvements seen in the diagonal layouts of Luke et al. [76], a second direct consequence of plague was a reduction in the influence of code growth on computational overhead. As code growth, or bloat, is known to increase over the course of a run in a variable sized representation (such as tree-based GP), the systematic elimination of individuals additionally implies less computational effort wasted toward evaluation of introns as evolution proceeds.

Smits and Vladislavleva adopt an Ordinal Optimization approach to algorithm design to address the computational cost issue through the modification of both population size and number of (random) fitness cases sampled during evolutionary runs under the regression domain [59] [107]. This design methodology is considered in conjunction with an archiving Pareto GP for the interesting case of linearly increasing subset size while systematically reducing population size. Evaluation using few exemplars over many individuals early in the run effectively corresponds to the use of a 'course' or 'soft' screen (goal softening), while performing exhaustive evaluation over a small population near the end of the run improves the screen fidelity over few alternatives. Results were indicated to be comparable to the standard approach, however with much lower variation [107].

2.5.2 Hardware Specific Speedups

As indicated in Chapter 1, of the five factors contributing to the computational overhead of evolving GP solutions (Equation 1.2), the inner evaluation loop is dominated by the number of exemplars over which the evaluation is made. The basic objective is therefore to conduct this evaluation as efficiently as possible through the use of an appropriate computer architecture. To this end, we briefly survey parallelization of the GP loop, specifically fitness evaluation.

A natural parallel approach involves a *control-parallel* system, having a main processor direct the evolutionary procedures of the algorithm (e.g., selection, reproduction), with distributed processors (nodes) bearing the parallel fitness evaluation duties. In practice, such an approach fits naturally with the steady state tournament-based selection, described in section, where the role of the main processor is to initiate selection and apply variation operators based on the results of a distributed tournament. In the parallel context a tournament round is initiated by the main processor, assigning individuals to processors (ideally on a one-to-one basis). The parallel step therefore solves the evaluation of individuals in parallel and returns the results to the main processor. The main processor completes the cycle with sequential application of genetic operators. An implicit assumption under the control-parallel model is global availability of training data, where this can be provided as a global read-only resource or as copies of training data distributed to each processor where a trade-off exists between storage requirements and inter-processor communication bandwidth. A critical disadvantage is that the maximum speedup due to the parallelization of the GP is in proportion to the tournament size which is frequently small.

A more common approach to the control-parallelization of GP is the use of distributed populations with migration, where this is has been described as the *asynchronous island* model. Under this configuration, GP sub populations are maintained at each node effectively parallelizing the entire algorithm at each processor. Following the (asynchronous) breeding and evaluation of the sub populations, a number of individuals are identified across all populations as immigrants, that will be incorporated into neighboring populations. Immigrants are identified probabilistically according to fitness and are queued at their destination nodes, and later assimilated when the

current (local) generational procedure has finished.

In general, the control-parallel models discussed above are readily amenable to low cost, commodity-class Beowulf clusters [7]; however such systems may be hampered by communication, storage and maintenance requirements as processors are added and as such they become more 'efficient' as the cost of fitness evaluation increases. That is to say as the ratio of fitness evaluation time to communication overhead (to control or master processor) decreases, the Beowulf model becomes increasingly efficient.

Finally, the concept of reconfigurable computing has been used to implement individuals as a Field Programmable Gate Array (FPGA) configuration for the purposes of evaluating individuals during fitness evaluation [63]. Needless to say, the number of function sets and problem domains to which this is applicable become increasingly limited given the overhead in converting code to hardware descriptions for 'programming' the FPGA.

2.5.3 Sub Sampling Algorithms

Previous work towards addressing the scalability of GP in software (given the large training set sizes of the classification domain) has broadly fallen under two paradigms: active learning and coevolution. These are related by their underlying approach to scalability (i.e., fitness evaluation over subset samples of the data) but differ considerably in their sampling mechanisms, where this typically involves addressing the appropriate proportion of data to be sampled from each class and sampling with sensitivity to maintenance of a relevant training gradient. These issues naturally have a significant impact on the performance of ensuing classifiers; Weiss and Provost, for example, demonstrated that sampling according to the naturally occurring class distribution provides best performance under an undifferentiated error evaluation, while a balanced sampling mechanism provides strong performance when an area under ROC (AUC) evaluation is considered [116].

Active Learning

The active learning paradigms, which include the Random Subset Selection (RSS) and Dynamic Subset Selection (DSS) algorithms of Gathercole and Ross [42] involve uniform stochastic sampling of the training data (in the case of RSS) and the use of age and difficulty heuristics to bias the stochastic sampling of training patterns during evolution (DSS). These algorithms have recently been extended to model hierarchical computer memories (hierarchical DSS [108] [21]) that involve multiple levels of selection. Such an approach is capable of handling very large data sets (those that exceed the conventional capacity of main memory), however these active learning frameworks generally involve manual selection of parameters such as block sizes and iteration limits. Moreover, these have only been demonstrated in the context of binary classification, although they are applicable to both supervised and unsupervised learning (thus discrete and real-valued cost functions) [117].

Closely related to these approaches are the Boosting and Bagging algorithms. Boosting and Bagging are two general purpose *ensemble* (team-based) techniques that are readily applicable to the GP classification domain with the potential for reducing computational cost while improving classifier accuracy. Under the Boosting algorithm, a series of classifiers are iteratively learned based on a biased sample of the data set. The sampling at each iteration is biased according to *weights* that are associated with each pattern being updated based on the previous iteration in proportion to misclassification. Specifically, exemplars that are incorrectly classified have their weights increased while correctly classified exemplars have their weights decreased so that each iteration focuses more on the previously misclassified exemplars. The final solution is therefore composed of many classifiers, with each assigned a weight in proportion to accuracy on the training data. Classifications are carried out (post training) by assigning unseen patterns to the class having the highest sum of weighted outputs. Owing to the tendancy to focus more on 'difficult' patterns, Boosting algorithms are susceptible to the over learning problem, however have the potential to improve classifier accuracy significantly [45].

Bagging (or Bootstrap Aggregating) involves the use of *bootstrap sampling* where data is sub sampled from the original set through uniform random sampling with

replacement. Each bootstrap sample forms the basis for training of a new classifier, with sufficient iterations yielding a collection of classifiers that employ a team-based voting policy post training. Each classifier is associated with a single vote and classifications are therefore carried out by assigning unseen patterns to the class having the highest vote total. In contrast to Boosting, Bagging is typically less susceptible to over learning since patterns are not sampled in proportion to difficulty; moreover, bagging typically results in a significant improvement in classification performance over a single classifier as the voting algorithm naturally contains the variance associated with individual classifiers.

A critical aspect of ensemble systems is disagreement among component classifiers [46], since combining classifiers that behave identically will clearly have no net improvement. Moreover, Krogh et al. [64] have formally shown that ideal Neural Network ensembles are composed of highly accurate classifiers with maximal disagreement, with this result being later confirmed empirically in terms of classifier generalization [92]. Therefore despite the general potential for performance (accuracy) increases through the use of ensemble methods, both Boosting and Bagging algorithms require an explicit specification of the number of component classifiers to employ; moreover each iteration requires a separate training run to generate the next classifier. These have additional implications for parameterization and computational overheads in the context of GP classification.

Coevolution

The second paradigm that has previously been considered in the literature with regards to scalability of GP as it relates to large data sets is that of competitive coevolution, where two (or more) populations interact (each changing in response to the other) to evolve simultaneously under the analogy of an adversarial game[4]. The central idea behind this paradigm is that the population contents provide the objectives for evolution which allows the potential for boundless training capacity in the absence of explicitly defined fitness objectives [3]. This is clearly advantageous

[4]It is also possible to phrase coevolution as a single population system, where interactions are defined by self-play [89]

where no fitness function is generally well known or such a function is difficult to define without *a priori* knowledge of the problem domain; this is typically the case in subset selection for training of a GP classifier. Specifically, one population will index the environment (e.g., a subset of indices to the larger data set or initial conditions within a test environment) whereas the second population represents the learners (in the context of the current work, GP classifiers).

Within the coevolution paradigm, two main approaches have emerged: the host / parasite model and the Pareto dominance model. The host / parasite (or predator / prey) approach models the competition between populations as a host / parasite interaction, where one population plays the role of parasite and the other is assigned the role of host [98]. Each population gains fitness at the direct expense of the other (the parasite receives fitness payoff for defeating the host, while the host is rewarded for successfully defeating the parasite). Such systems have the potential to engage in the evolutionary equivalent to an 'arms race' [100] [3], however the dynamics of these systems are infamous for their oscillatory behavior and fragility during training [17]. Specifically well-studied problems include disengagement (leading to loss of gradient), relativism, the red queens effect, and intransitivity [17] [115] [56]. Several approaches have been proposed in the literature for correcting these problems including: moderation of parasite virulence [17], the 'hall of fame' memory mechanism, competitive fitness sharing, and shared sampling selection [100]. A competitive coevolution (host / parasite framework) has been previously examined for GP classification where the host population was configured to represent classifiers while the parasite represented subsets of the training data [81]. This configuration was seen to suffer from a high degree of sensitivity to virulence parameters and oscillatory training dynamics, although training was achieved on data sets having up to eleven thousand exemplars.

The Pareto coevolution model takes the approach of 'opponents as objectives' [24], combining the notion of coevolution with techniques from the EMO literature, including sorting and ranking based on the concept of Pareto dominance [44]. Classically this has been employed in the GA (in particular, self-play and gaming) context [89]. Recently Pareto coevolution has become the subject of considerable interest

as the relevance of 'distinctions' made by opponents (or 'tests') has been demonstrated as the basis for maintaining an appropriate training gradient (also known as engagement) relative to the coevolving population (of 'learners') [34]. Under the DELPHI (Dominance-based Evaluation of Learners on Pareto Hillclimbing Individuals) algorithm [24], defeating a learner is not necessarily the most valuable result of an interaction; rather the ability of a test to provide a new distinction between learners is rewarded; this provides the basis for progress in the evolution of learners (i.e., a training gradient). Several authors have since proposed competitive coevolutionary models based on the ability to distinguish between learner behavior, including the use of 'Test Banks' in [14] and [35]. In particular, the Incremental Pareto Coevolution Archive (IPCA) algorithm [23] (which extended the DELPHI algorithm with archive mechanisms, thereby guaranteeing monotonic progress in learning) has recently lead to the Pareto coevolution model being extended to the binary classification context for GP in the form of the Pareto GP Classifier (PGPC) algorithm [70] [71]. The significance of this is that by associating tests with training patterns and learners with GP classifiers, the resulting framework coevolves a compact subset of objectives (tests) on which to train learners (classifiers), thus providing an automatic sampling mechanism which achieves scalability in the GP training algorithm without recourse to hardware-specific speedups or the conventional active learning mechanisms. The ensuing solution takes the form of the Pareto front of learners with a post-training voting policy acting as the mechanism to provide class labels on unseen data. As well be demonstrated, such an approach results in solutions that adopt an 'ensemble' or 'weak learner' pattern of coverage, with learners from the Pareto front responding equally to the majority of exemplars. We attribute the low distinctiveness of the solutions from the competitive coevolutionary model to the use of a global wrapper function and single objective fitness function (classification count).

2.6 Discussion

The case was made in the Introduction that the number of fitness evaluations necessary to evolve GP classifiers under generic problem domains consists of five factors: number of classes, number of initializations, population size, generation limit, and

the training exemplar count. The current work further establishes the Pareto coevolution model under a multi-class framework for classification in a multi-objective GP, where the solutions are explicitly encouraged to take the form of collaborators which naturally decompose classification problems, thereby providing the basis for highly modular and compact solution sets over multiple classes as the end result of a single trial of GP. Our solution to this problem has four central components:

1. Local membership or wrapper function : Establishes a novelty detection model of operation as opposed to the typically employed discriminant model of machine learning. In doing so, we are able to provide much more predictable behavior under unique exemplars than is the case under the discriminant model.

2. Evolutionary multi-objective optimization : The utility of multiple objectives enables us to build models that are parsimonious (therefore simple to evaluate) as well as providing the opportunity to focus the goal of the classifiers more effectively. In particular, when used in combination with the local membership function, we will be able to provide a very clean model for problem decomposition and early stopping.

3. Competitive coevolution : Scaling to large data sets relies on assuming a competitive coevolutionary model. In particular, we focus on the Pareto test-based paradigm exemplified by de Jong's IPCA model in preference to host-parasite or active learning algorithms, although any of this family of algorithms would be equally applicable.

4. Default sampling heuristic : Although competitive coevolution provides a mechanism for identifying the most useful exemplars for driving the evolution of a learner population it says nothing about how to sample the original set of training exemplars. In recognition of this we will adopt the balanced uniform sampling algorithm benchmarked by Weiss and Provost, where this has been empirically shown to provide a a good basis for building 'robust' classifiers under the decision tree induction ML paradigm [116].

Our design additionally assumes a means to explicitly encourage the EMO model to return a front in which individuals explicitly co-operate to return a set of classifiers

with unique decompositions of the problem domain, as opposed to merely trading off different objectives. The design of algorithms appropriate for realizing these objectives is presented in Chapter 3.

Chapter 3

Algorithms

In order to develop the proposed approach to GP classification we will introduce the framework in terms of the high level pseudo code listing provided in Figure 3.2. This listing describes the proposed framework as a seven step algorithm having two basic components: a cooperative coevolutionary training section (steps 3-4, Figure 3.2) and a competitive coevolution archiving section (step 5, Figure 3.2). These two components of the framework will be principal subjects of discussion as the chapter develops, however we begin by describing the basic model of GP classification assumed in the current work. This is followed by a general discussion of the high level pseudo code provided in Figure 3.2. We finally provide an explanation of the framework's organization and data flow before proceeding with a detailed presentation of the underlying algorithms. The chapter concludes with a short summary of technical benefits including a computational complexity analysis.

3.1 Classification Model

An interaction between a GP individual and a single exemplar is defined by executing the GP program using the exemplar features as arguments. The result of a single program run on an exemplar (an interaction) is a real-valued output, mapping the exemplar (from a typically multi-dimensional feature domain) to a one dimensional number line that we refer to as the 'GP Output' space, or simply *gpOut*. An example of this process is provided in Figure 3.1, where the mapping process is repeated to map n exemplars to n points on *gpOut*. Such a mapping alone, however, conveys no class label information; yet the goal of a GP individual under the classification context is to provide an expression mapping the input space (of the problem domain) to a class consistent output label.

Canonical GP employs a global membership function (wrapper operator) to impart

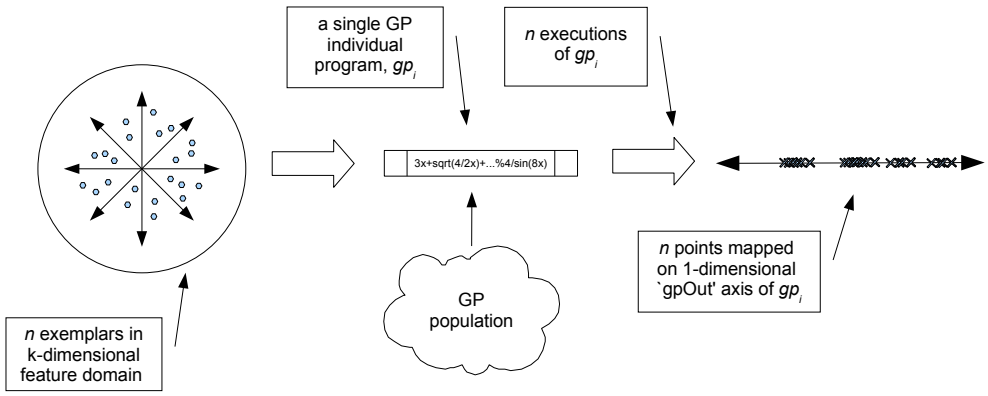


Figure 3.1: Basic mapping process of GP classifier model.

class labels. Such an approach effectively casts all points on the *gpOut* axis that are greater than (less than) zero as in-class (out-class) exemplars, and counts the number of misclassifications by way of a cost function. We maintain that such an operator also enforces the 'super individual' approach to problem solving with little support for problem decomposition. Moreover, this implicitly assumes that partitioning the class labels about the origin of the *gpOut* axis is appropriate for the problem domain in question; however, such an assumption is not likely to be true in practice [75]. Under the current model, label assignment is therefore realized by means of a local membership function (LMF) that explicitly assigns classes to different regions of *gpOut*, suggesting that the mapping should preserve class consistency, i.e., that similar exemplars (e.g., within each class) should be mapped to the same local neighborhood on the *gpOut* axis.

The basic features of our Competitive Multi-objective Grammatical Evolution (CMGE) classifier are now summarized as follows relative to the pseudo code listing provided in Figure 3.2:

1. Identification of the subset of exemplars over which individual (learner) evaluation will take place (lines 2a, 2b);

2. Identification of the local membership function (LMF) for each individual, relative to the associated *gpOut* distribution (lines 4a - 4d);

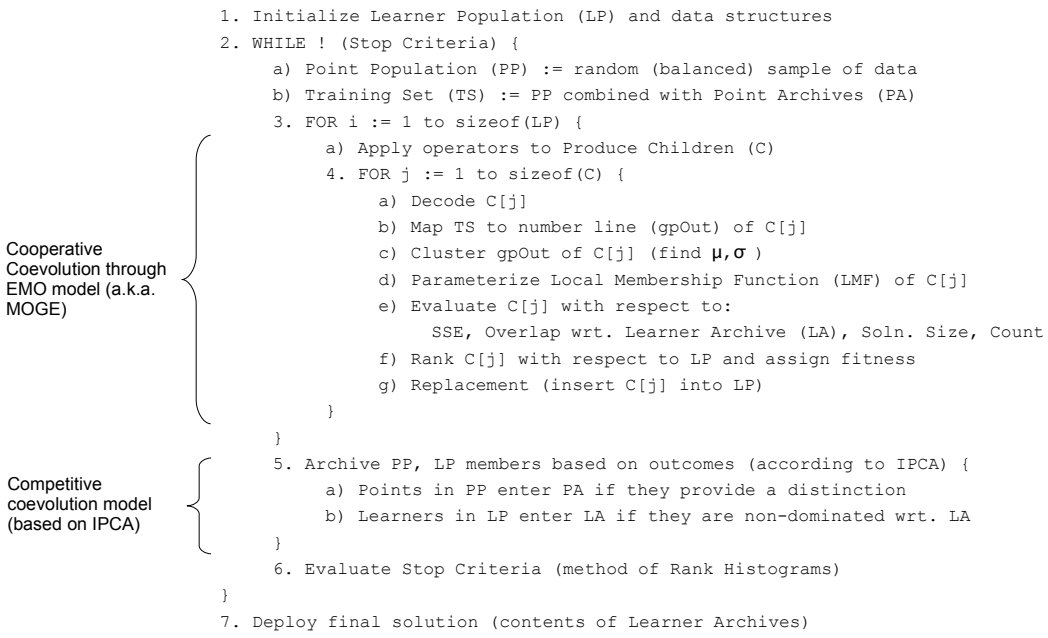


Figure 3.2: High-level pseudo code listing.

3. Evaluation of individuals relative to the learning objectives under a multi-objective methodology (lines 4e - 4g);

4. Identification and archiving of the most valuable individual classifiers and exemplars (lines 5a, 5b);

5. Class-wise assessment of early stopping criteria (line 6).

With respect to each of these features, we provide the following introductory comments prior to highlighting the details of the pseudo code listing:

Feature 1 : A class-wise random balanced sampling heuristic is enforced such that each class has equal representation in the point population. This policy is assumed as individuals (in the point population) represent exemplar indexes. Such a representation means that there is no structure on which to build search operators. Moreover, structure could not be derived without recourse to an appropriate pre-processing activity such as clustering or graph theoretic models, both of which have considerable computational overhead;

Feature 2 : The local membership function is derived from the distribution of points on the *gpOut* axis. As such, no attempt is made to incorporate the concept of class labels when deriving the location of the local membership function. Instead, we assume a that the local membership function is expressed by a Gaussian, the parameters of which are derived by first applying a clustering routine to the individual's point distribution on the *gpOut* axis. Having identified the subset of points associated with the most dense cluster the mean and variance of the Gaussian local membership function for that particular individual are established (Figure 3.3). Only the subset of points within the cluster as evaluated on a specific individual's *gpOut* axis are associated with the local membership function. This is the first property by which we establish problem decomposition;

Feature 3 : At this point we have a set of individuals with their corresponding local membership functions and therefore possibly unique subsets of exemplars established. Performance evaluation may now take place in two stages. Class label associated with the individual is established by assuming that the individual takes the label of the exemplar with maximum membership of the Gaussian. With the introduction of class labels we may evaluate performance over multiple objectives, albeit for the subset of exemplars actually mapped to the Gaussian alone (Figure 3.4). Moreover, the multi-objective view provides the opportunity to reward non-overlapping behaviors as well as error minimization. Naturally, having established the relative fitness of individuals (Figure 3.5), selection and reproduction takes place under a Pareto multi-objective model which encourages diversity without recourse to distance based metrics [66]. This completes the explicitly co-operative aspect of the framework.

Feature 4 : Competitive coevolution is now used to identify the best points and classifiers to retain outside of the point and learner populations. To do so, an outcome vector is constructed over the current contents of the point archive and population, relative to the current contents of the learner archive and population. As such this process follows the IPCA algorithm of de Jong [23], however,

any form of this class of competitive coevolution would be appropriate. Novel additions to the IPCA approach include support for class-specific archives in both the points and learners, finite archive sizes, and therefore the adoption of suitable pruning heuristics to enforce the finite archive sizes. Moreover, special attention is necessary to the derivation of an appropriate mechanism for establishing the outcome vector. In particular this is a real-valued pairwise matrix of the behavior of each individual relative to points (Figure 3.6). Only individuals that are non-dominated in the Pareto sense (with respect to outcome vectors) represent candidates for archiving. Similarly, only the points making the distinction between dominated and non-dominated learners represent candidates for the point archive.

Feature 5 : Stop criteria is established in a problem independent manner by making use of the concept of Pareto rank histograms, as established in the Pareto multi-objective technique adopted in Feature 3. Unlike the original GA context in which this concept was derived we also deploy it in a class wise manner. This enables us to declare classes converged class-by-class, thus providing the ability to redeploy the individuals associated with that class, such that they are reassigned to classes as yet not converged.

3.2 High Level Algorithm Discussion

The standard initialization process of line 1, Figure 3.2 randomly sets all GP population members (learners) and prepares the relevant data structures, including memories (archives) for both learners and exemplars (data points or simply, points). This is discussed in detail in Section 3.6.1. A while loop (line 2, Figure 3.2) encloses the main sections of the algorithm, ensuring that the steps of the body (i.e., the training of GP) are repeated until stopping conditions are met (as evaluated at the end of each iteration in line 6, Figure 3.2). Steps 2a and 2b of Figure 3.2 set up the training set at each iteration ensuring a balanced view of data, thus enabling robustness against problems having unbalanced class distributions. This process is associated with Feature 1 (Section 3.1) and its underlying motivation is discussed in detail in Section

3.7.

Line 3 of Figure 3.2 begins the cooperative coevolution training loop which employs an EMO model loosely based on that of Kumar and Rockett [66] to train GP. This is discussed in detail in section 3.7.2. On each pass of the loop, selection and variation operators are applied to the GP population and children are produced (line 3a, Figure 3.2). Next, individuals are decoded to program form (line 4a) and the current selection of exemplars are mapped to the *gpOut* axis, as indicated in Figure 3.1. We now require a mechanism to identify the local membership function (LMF) neighborhood without resorting to inappropriate or arbitrary predefinitions of regions within the *gpOut* axis. Once such a neighborhood has been defined we can then attach an evaluation function to the members of the neighborhood, incorporating the behavior of other individuals in order to encourage co-operative approaches to problem solving. In order to achieve this goal we assume that the neighborhoods of most relevance are those that have the highest density (see Figure 3.3). Note that this requirement is relative to the distribution of points on the *gpOut* axis, and is independent of class label, the latter property being enforced later by way of the fitness function.

At this stage (line 4c of Figure 3.2) we have the basic requirement for a clustering algorithm to be applied to the subset of points identified by the competitive coevolutionary model as mapped to each individual's *gpOut* axis. The clustering algorithm returns the location of the mid point associated with the 'most dense' set of points, μ, and exemplars associated with this cluster, M. We now have the properties for the local activation function defined in terms of a Gaussian with mean μ and variance σ (the latter estimated over the points associated with the cluster) and parameterization proceeds on line 4d of Figure 3.2. A generic example of steps 4c and 4d (corresponding to Feature 2 of Section 3.1) is provided in Figure 3.3. A more detailed discussion of this process is provided in Section 3.7.3.

With the properties for the local membership function of the GP mapping established we now introduce exemplar labels and apply the multi-objective fitness criterion (lines 4e and 4f of Figure 3.2, or Feature 3 of Section 3.1). The objectives are designed to encourage: least ambiguity in cluster membership, non overlapping

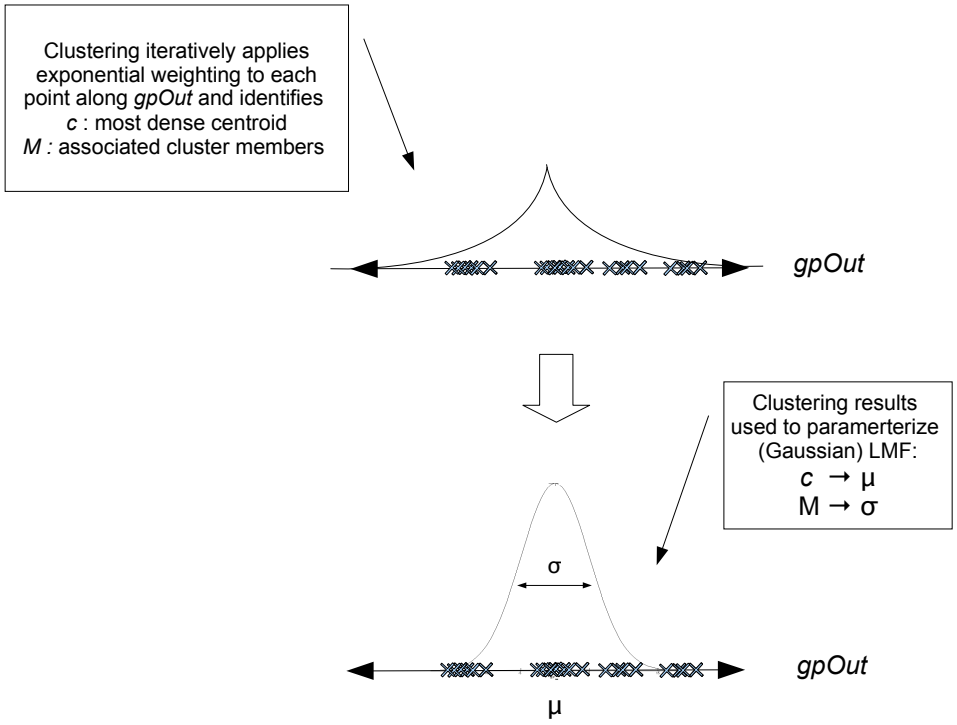


Figure 3.3: Application of clustering to establish parameters for local (Guassian) membership function.

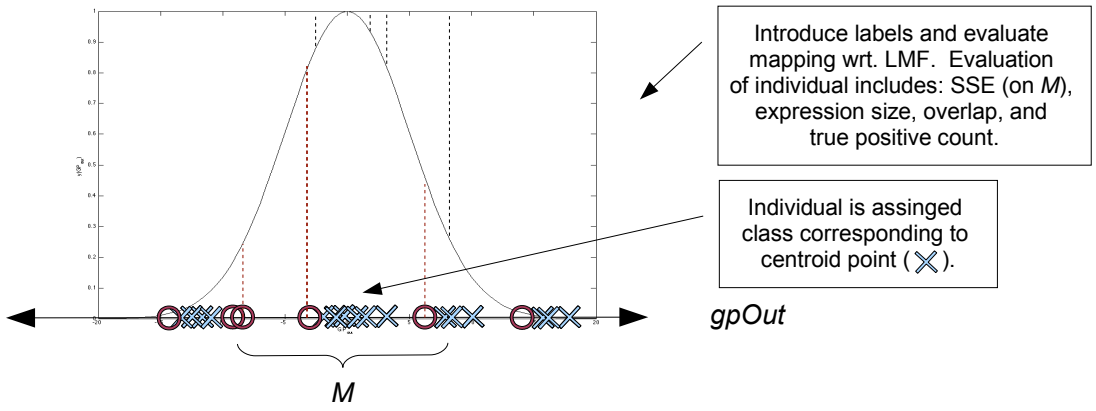


Figure 3.4: Introduction of pattern labels to evaluate individual mapping with respect to local (Guassian) membership function.

behavior of the exemplars mapped to different individuals, maximization of the number of in-class exemplars mapped to an individual, and simplicity of the GP mapping. A basic illustration of the error evaluation process, for example, is provided in Figure 3.4: the GP individual adopts the class label associated with its centroid and is now evaluated as a subspace (a one dimensional number line, *gpOut*) and associated with in and out of class exemplars.

Note that, in common with the findings of other EMO research, we establish a set of objectives that have a degree of implicit 'tension' between them [20]. In doing so we are able to encourage mappings that reduce the likelihood of degenerate solutions. Moreover, in order to measure these objectives, the mapping is assigned a class, where this is assumed to correspond to the class of the point at the center of the local membership function. In taking this route we avoid making any assumptions regarding which individuals are mapping which classes, and effectively let individuals compete for the right to map exemplars. In particular the objective of minimizing the number of (in-class) exemplars shared between different individuals encourages diversity in the GP mappings, helping to ensure that we do not evolve populations that concentrate on mapping the 'easy' exemplars at the expense of the 'difficult'.

A detailed discussion of the motivation behind our choice of objectives and their associated evaluation is provided in Section 3.7.4.

Our variant of the PCGA EMO algorithm enables fitness assignment on the basis of pairwise Pareto ranking along the four objectives and enforces replacement of the lowest ranked population members on lines 4f and 4g of Figure 3.2. A basic example of the Pareto ranking (ranks are directly related to fitness score (see Section 3.7.5)) is illustrated in Figure 3.5. The significance of the Pareto ranking and ensuing fitness assignment is that selection operators proportionately favor individuals of higher fitness (lower ranking) over those having lower fitness (higher ranking). This tends to encourage the GP algorithm to more frequently sample material corresponding to individuals that lie closer to the Pareto front, in hopes of evolving improvements in the chosen objectives.

Figure 3.5 also introduces the concept of a rank histogram, which essentially summarizes the content of the population (in objective space) in terms of the Pareto ranks so that content can be readily compared between training epochs. When calculated for each class, this provides the basis for Feature 5 (Section 3.1), the evaluation of early stopping (line 6 of Figure 3.2). This is discussed in detail in Section 3.9. In essence, classes associated with unchanging content are assumed to have converged [65]. At this point resources ('free' individuals) are redeployed to the classes that have not yet converged. Such an approach is naturally more robust than attempting to set error thresholds which tend to be data set specific.

The above process (steps 3 and 4 of Figure 3.2) define the cooperative EMO model. This portion of the main loop is performed in combination with the competitive model (a variant of de Jong's IPCA algorithm [23]) for the purpose of adapting learner and test point archives as memories at line 5 of Figure 3.2. That is to say, the competitive coevolution model's evaluation is conducted over the contents of the subset of training exemplars (line 2b of Figure 3.2) dynamically identified by a competitive co-operative model for archiving the most discriminatory test points (step 5a) and non dominated learners (step 5b), both from the perspective of a Pareto front (see Figure 3.5). The competitive model thus plays a primarily archival role (Feature 4 of Section 3.1), acting as a memory for the cooperative model. The archive entry criteria are evaluated

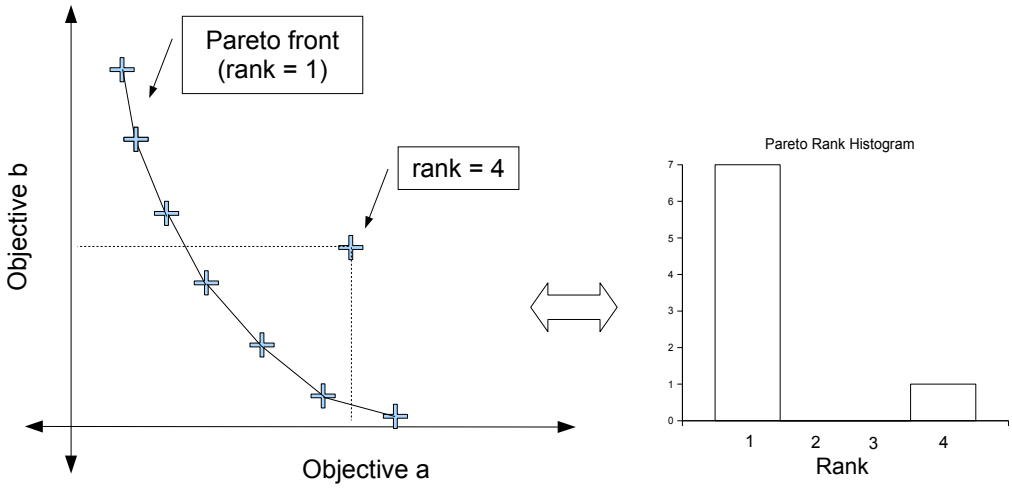


Figure 3.5: Pareto ranking of individuals and relationship to early stop criteria.

in terms of GP classification 'outcomes' (see Figure 3.6) which are directly related to the LMF definition and it's associated performance on the training set. A detailed description of the competitive archiving model for points and learners is provided in Section 3.8.

Deployment of the classifier (step 7, Figure 3.2) takes the form of copying the contents of the learner archives and assignment of weights to each on the basis of the training data (see Section 3.10). A winner-take-all policy with respect to LMF outputs determines the assignment of class labels among the team individuals.

In order to develop the above model in detail we incrementally add detail to the above high-level motivation, with Section 3.3 introducing the architecture for EMO and competitive coevolution. Section 3.4 presents the algorithm 'road map' with Sections 3.5 to 3.11 stepping through the ensuing algorithmic details. Section 3.13 performs a complexity analysis of the resulting algorithms with Section 3.12 summarizing the chapter.

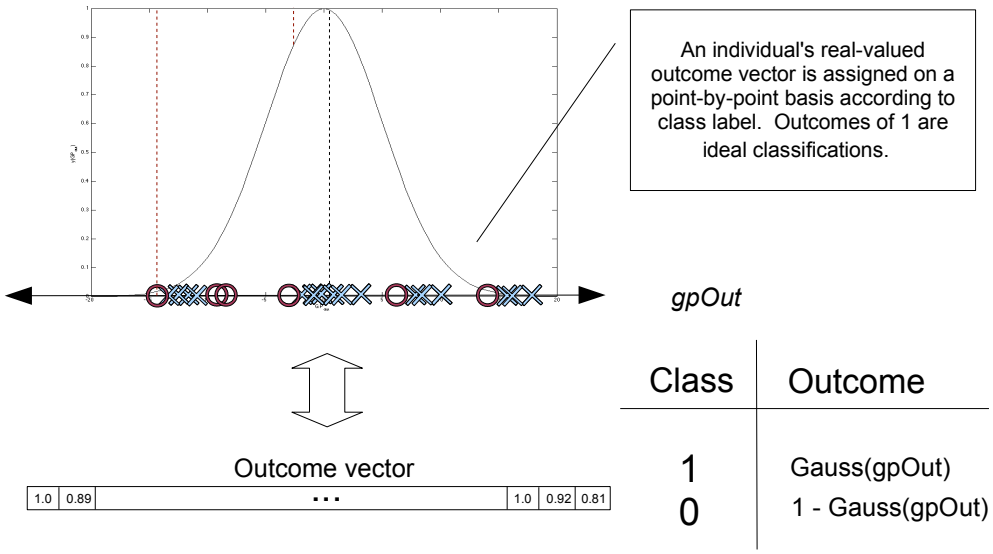


Figure 3.6: The 'outcome vector' links the framework into the competitive coevolution model which provides memory through point and learner archiving.

3.3 Framework Organization and Data Flow

In the following we adopt the terminology of the Pareto-coevolution literature and the term *test* or *point* will be used interchangeably in reference to a single case, exemplar or data instance, while a *learner* refers to a single individual or GE expression. Moreover, we distinguish between the *training set* (TS) and the *training data* (TD), with the former referring to the union of point archives with the point population and the latter referring to the entire collection of training exemplars.

The Competitive Multiobjective Grammatical Evolution (CMGE) framework employs two tightly integrated algorithm stages. The first is associated with the Multiobjective Grammatical Evolution algorithm (or MOGE as described in [82]) and the second is an IPCA-based archiving algorithm, concerned with retaining appropriate learners and training points at the conclusion of each learning cycle. Relationships between the various components of the framework are outlined in Figure 3.7. Rectangular components with vertical stripes on opposite sides represent functions or processes, while parallelograms represent data structures supporting either learner or point populations. Lines between components indicate an exchange of data and arrows are used to represent the direction of data flow. The broken horizontal line

separating components 1, 2 and 3 from 4, 5 and 6 in Figure 3.7 illustrates the logical separation of the system into the MOGE / IPCA stages. Figure 3.7 will be used to discuss the CMGE framework in high-level terms in the following two sections before focusing on the algorithm details in Section 3.4.

3.3.1 MOGE Component

At a high level, MOGE (component 1 in Figure 3.7) employs a training set composed of a sample of exemplars (known as the point population, component 2) that are selected from the main data set at random, subject to a class-wise balance requirement as motivated by the findings of Weiss and Provost [116]. These exemplars are augmented with several additional sets of points that are selected by IPCA for their class-wise relevance to training and are known as the point archives (component 6 in Figure 3.7). The training set, indicated in component 2 of Figure 3.7, is the union of the point population with the point archives and represents a non-repeating subset of the underlying data. Moreover, this subset provides exemplars for tests that are currently supported by the learner archive (i.e., a memory of what has been learned) as well as for uniformly sampled tests from the original data, with the goal being to continually drive incremental evolutionary improvements. On each learning cycle, MOGE utilizes the updated training set to evolve learners, modifying the learner population in component 3 of Figure 3.7 through selection, mutation and recombination while favoring the propagation of material from fit individuals evaluated according to the multi-objective framework. This evaluation process has a cooperative aspect as it is partially based on feedback from learner archives as discussed in the next section.

3.3.2 Coevolutionary Component

At the conclusion of each learning cycle or *epoch*, control is passed to the IPCA-based archiving process indicated by component 4 in Figure 3.7, where the current point and learner populations are evaluated against class-specific archive entry criteria. Learner and point archives are indicated by components 5 and 6 in Figure 3.7, respectively. Learners satisfying the learner archive entry criteria are committed to the archive corresponding to their class, while accepted points are committed to the point archive

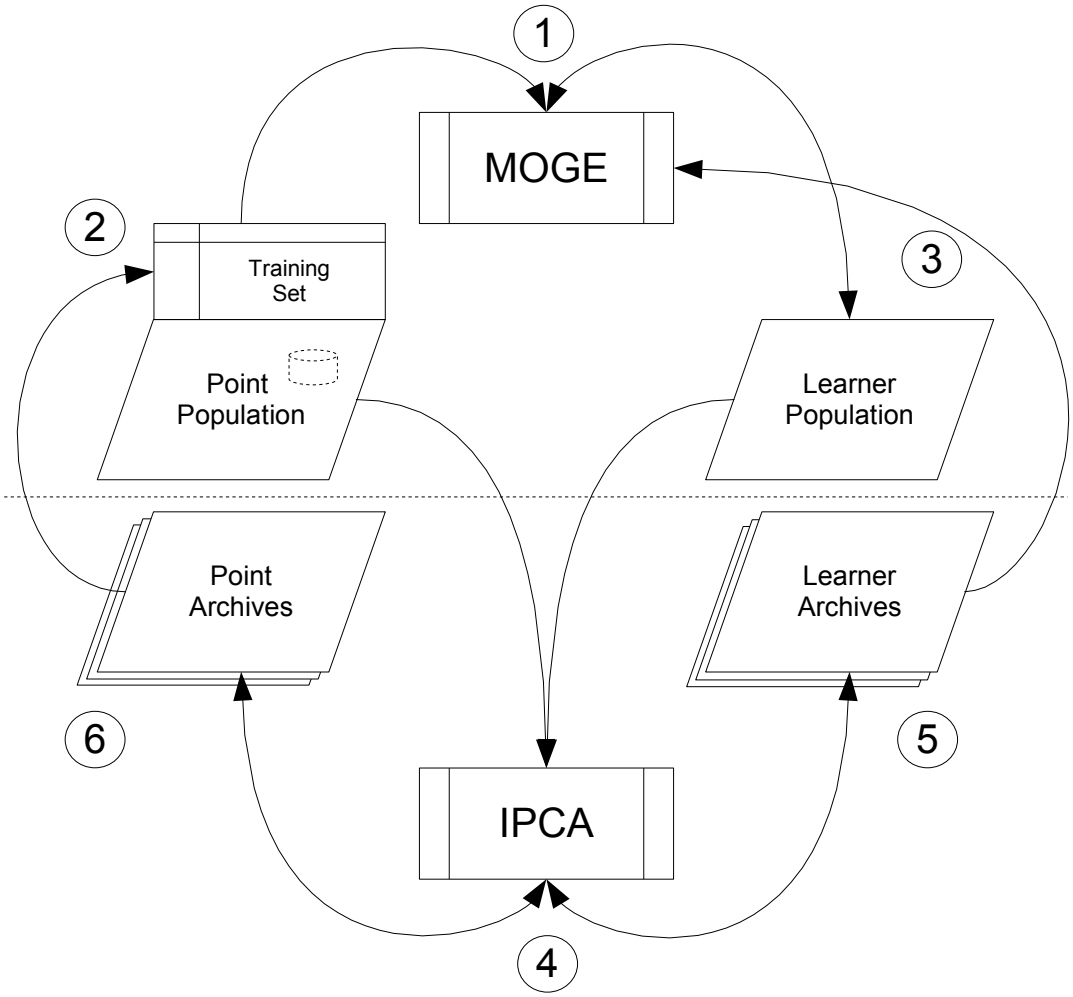


Figure 3.7: High-level CMGE framework and data flow.

corresponding to the class for which they have been identified as having learning significance. Point archives are used to build the training set for the next cycle of learning, while the coverage behavior of the learner archive individuals partially directs the multi-objective evaluation of the learner population during the next cycle of evolution. The final solution is taken as the learner archive contents.

The use of a Pareto coevolutionary archiving mechanism in the CMGE framework was motivated by the work of Lemczyk et al. [70] [71] (work which was originally based on de Jong's Incremental Pareto-Coevolution Archive (IPCA) algorithm [23]). The archiving framework presented here, however, extends beyond Lemczyk's binary context to multi-class classification, where separate learner and point archives are introduced for the purpose of supporting learning on the basis of class-appropriate objectives. Moreover, unlike Lemczyk, we make use of multiple archives with early stopping criteria based on the behavior of the class-wise Pareto fronts (Algorithm 3 line 5) enabling class / problem specific stop points to be identified [66]. The notion of an outcome (and therefore the learner and point archive entry criteria) has also been reformulated to take advantage of the current framework's real-valued outputs, indicating classification certainty (degree of LMF membership) as opposed to purely binary values (correct vs. incorrect). Basing archive entry criteria on this approach is specifically intended to promote team-oriented behavior and will be discussed in more detail in Section 3.8 and Chapter 8. Finally, the learner archive contents are fed back into the evolutionary (MOGE) loop to encourage a cooperative search, where newly evolved individuals are rewarded for providing classifications that have not already been learned, relative to the learner archive content. These contributions combined with the original MOGE algorithm result in the ensuing CMGE framework, permitting multi-objective evolution of classifiers based on local (Gaussian) membership functions, with the Pareto competitive archive model naturally reducing the number of exemplars necessary to perform fitness evaluation as well as providing class wise early stopping, archiving according to team specific criteria and cooperation in the multi-objective evolutionary search.

3.4 CMGE Algorithm Descriptions

The CMGE framework presented in this thesis is detailed by the series of algorithms in the following sections. All learners are based on the same context free grammar (CFG), G (although the contributions of this work are independent of the particular GP framework assumed). High level algorithms described by:

- Algorithm 3 (cmgeMain): expressing the sequence of high level function calls;
- Algorithm 4 (initCmge): detailing the global process of initialization;
- Algorithm 6 (trainCmge): the process of training the CMGE;
- Algorithm 25 (ipcaEvaluation): detailed criteria by which IPCA is applied to direct training;
- Algorithm 32 (stopCmge): summarizing the early stopping criteria;
- Algorithm 34 (buildSolution): assembles the final classifiers (post training) from the learner archives.

Each of these high level algorithms is assumed to have global read / write access to all variables listed in Table 3.1; this will be indicated by a parameter list containing ellipsis (...). All other algorithms assume data passed by value unless otherwise noted in the algorithm input / output descriptions. Access to a function named 'class', which returns an exemplar label or individual class assignment, is also assumed along with array-based functions: max, min, sort, sum and mod, which perform traditional numeric roles.

As outlined in Table 3.1, the high-level algorithms employ $2c+2$ populations and archive variables, where c is specified by the number of classes in the problem. That is, separate point and learner archives are retained for each class. This ensures that point archives are able to retain the most appropriate tests (training points) for learners associated with each class. The descriptions that follow employ several additional data evaluation structures which are also provided in Table 3.1. All parameter values associated with the following algorithms are specified in Table 4.5; however, no

Table 3.1: Algorithm Data Structures and Parameters

Description	**Abbrev.**	**Num**	**Capacity**
Populations and archives			
Point population	PP	1	PP_{size}
Learner population	LP	1	LP_{size}
Point archives	PA	c	PA_{size}, ea.
Learner archives	LA	c	LA_{size}, ea.
Labeled Classification Data			
Training data	TD	1	N
Training set	TS	1	$PP_{size} + c \cdot PA_{size}$
Evaluation structures			
Non-converged classes	NC	1	c
Useful tests	UT	1	PP_{size}
Undefeated tests	UT^*	1	PP_{size}
Useful learners	UL	1	LP_{size}
Non-useful learners	$\overline{UL}$	1	LP_{size}
Undefeated learners	UL^*	1	LP_{size}
Current rank histograms	RH^t	c	LP_{size}
Previous rank histograms	RH^{t-1}	c	LP_{size}

attempt was made to optimize these selections. The over-riding interest is to provide uniformity across the data sets utilized in the ensuing benchmarking activity, Chapter 4.

The highest level function (cmgeMain, Algorithm 3) is described first in Section 3.5. Sections 3.6 to 3.10 provide detailed descriptions of the basic steps of Algorithm 3. Where appropriate, CMGE design decisions will be highlighted with particular attention paid to the fundamentally novel aspects of the algorithms.

3.5 Main Function

The CMGE main function is provided in Algorithm 3, which exhibits the three high-level blocks of functionality: initialization (line 1), the outer training loop (lines 2 - 6) and post-training solution assembly (line 7). These will be discussed with increasing detail in the following sections. Line 1 (initCmge) carries out the initialization of all high level variables including populations and data structures supplied in Table 3.1. This is examined in more detail in Section 3.6. Lines 2 - 6 enclose the outer training

loop which cycles until a stop criterion is reached. Stop criteria are discussed in further detail in Section 3.9. Line 3 of Algorithm 3 is the call to the high-level learning cycle, trainCmge, an earlier version of which was known as MOGE [82], that provides the multi-objective framework for training the classifiers. Adopting a multi-objective scheme enables the user to encourage, say, simple as well as accurate classifiers. Moreover, as a natural consequence of the EMO paradigm, solutions may take the form of more than one classifier for the same class [82]. This is the subject of discussion in Section 3.7. Line 4 of Algorithm 3 addresses the computational overhead of fitness evaluation through a Pareto (competitive) coevolutionary algorithm that is able to identify minimal sets of training examples to conduct training while maintaining a memory of the most relevant learners. A detailed discussion of archiving properties and the ipcaEvaluation function in general are provided in Section 3.8. The final step of Algorithm 3 is line 7, which chooses solutions to be included in the final classifier and assigns a weighting to each member that is later used to combine the various member outputs to a single output per exemplar.

Algorithm 3 cmgeMain(...) - High level algorithm for the CMGE framework.

Input: Assumes global read / write access to variables described in Table 3.1.
Output: Final classifier (solution) S.

```
1: initCmge( ... )
2: while ! Stop criteria do
3:    trainCmge( ... )
4:    ipcaEvaluation( ... )
5:    stopCmge( ... )
6: end while
7: S := buildSolution( ... )
```

3.6 Initializations

Initialization proceeds as indicated in Algorithm 4. This algorithm has two roles, firstly to initialize the point population PP and learner population LP on lines 1 and 2, respectively. Secondly, the class-wise archive structures LA, PA are initialized on lines 4 and 5. The point population is initially assigned to the empty set on line 1; however, this population is required to maximize diversity, hence will be reinitialized

Algorithm 4 initCmge(...) - High level CMGE algorithm initializations.

Input: Assumes global read / write access to variables described in Table 3.1.
Output: Initialized structures.

```
1: PP := {∅}
2: LP := initLearners( LP )
3: for i := 1 ... c do
4:     LA[i] := {∅}
5:     PA[i] := {∅}
6:     NC[i] := i
7: end for
```

after each training epoch, Section 3.7. The learner population initialization of line 2 is carried out by a call to Algorithm 5, initLearners and is discussed in Section 3.6.1. Archives (LA, PA) may only accept individuals satisfying the Pareto dominance criteria, Section 3.8, thus initially contain no individuals. Finally, line 6 of Algorithm 4 represents the initialization of the data structure (NC) used to detect early (class-wise) convergence.

Algorithm 5 initLearners(L) - GE individual initializations.

Input: Set of GE learners L to be initialized to legal expressions.
Output: Initialized GE learners.

```
1: for i := 0 ... |L| − 1 do
2:     repeat
3:         for j := 0 ... CODONS-1  do
4:             L[i].codon[j] := random( {0, 1, ... MAX_CODON_VAL} )
5:         end for
6:     until  geMap( L[i].codon )
7: end for
8: return  L
```

3.6.1 Initializing Learners

A set of GE-based learners L is initialized by Algorithm 5. Lines 2 - 6 perform the initialization over all learners in the population, with each initialization being repeated until a legal mapping (return value of 1) is produced by geMap (Algorithm 2) on line 6. When this process is successful it ensures that no degenerate GE individuals are defined in the learner set [91]. Line 4 of Algorithm 5 assigns codon (gene) values over

the length of the individual (CODONS), with uniform probability over the range [0, MAX_CODON_VAL]. During initialization, there are no requirements on expression length, however the genotype is limited in the number of rule selection values (by the CODONS parameter) and expressions are constrained to a maximum string length of MAX_EXP_LEN (see Table 4.5).

3.7 Training

The main training function is provided in Algorithm 6, trainCmge. At the outset of each epoch, the point population (PP) is filled using random uniform selection without replacement over the range of all training pattern indices by Algorithm 7, fillPtPop, on line 1. The point sampling aspect of the framework is discussed in further detail in Section 3.7.1. In line 2 of Algorithm 6 the training set TS for the current epoch is the union of all point archives $PA[1] \dots PA[c]$ with the point population PP. Notably all point archives contribute to TS, regardless of any prior class convergence. This ensures that entire classes of points cannot be ignored once they have converged due to class-specific stopping criteria discussed in Section 3.9.

In line 3 of Algorithm 6, the learner archives LA are evaluated against the current training set to establish their current coverage, which will be used to encourage cooperative behavior among learners in this respect. That is, the coverage of the learner archives, as determined here, will be used to evaluate an explicit coverage metric on each of the current and newly evolved individuals, favoring individuals with low overlap (see Section 3.7.4). The next step of the training algorithm, line 4, provides a population-wide evaluation of fitness using the multi-objective paradigm across the learner population (LP). At this step, an local (Gaussian) membership function is assigned (or re-assigned, after each archiving cycle) and a Pareto evaluation of individuals on the multiple objectives (see Section 3.7.4) provides a scalar ranking of that is used to establish fitness for selection (Section 3.7.5). The multi-objective evaluation, ranking and fitness assignment are discussed in greater detail in Section 3.7.4.

Lines 5 - 20 of Algorithm 6 constitute our original MOGE learning cycle. The basic procedure involves fitness proportionate, stochastic selection (Algorithm 8, select) on

lines 6 and 10 defining parents $p1$ and $p2$, followed by application of the genetic operators: crossover (Algorithm 9, applyXover) on line 14 and mutation (Algorithm 10, applyMutation) on line 17. The while loop at line 9 prior to the second parent selection is intended to provide a sample of size c (number of classes in the problem) to find a class match for the first parent selected in line 6; failing this, however, a fitness proportionate random second parent (irrespective of class) is chosen to avoid a more costly search.

Following the variation blocks, the newly created individual is next passed to the evalLearners function (Algorithm 12), on line 19 where the LMF assignment and multi-objective evaluation are carried out. Finally the new individual is inserted into the population with the replace function, Algorithm 22 on line 20. This learning process is iterated until the equivalent of an entire population has been generated, however the replacement scheme itself is not generational; rather a simple policy of replacing of the lowest ranked learner is enforced in Algorithm 22, replace. While this does not guarantee monotonic progress of the MOGE Pareto front it does provide an elitist approach to replacement that allows for diversity in the classifications which can be favorable in the context of the later archiving stage, where monotonic progress of candidate solutions is guaranteed [23]. The learning cycle is discussed in detail below, in Section 3.7.2.

3.7.1 Generating the Point Population

The point population plays an exploratory role in the coevolution process by sub sampling and maintaining a balanced view of the training data from the perspective of the learners. Such a view is taken as a consequence of the findings by Weiss and Provost as discussed in Chapter 2 regarding the general robustness of such a sampling policy with respect to several classifier performance metrics [116]. The balanced view requires that an equal number of points from each class is represented in the point population. Specifically, an equal allocation is made for each class and the point generation function fillPtPop of Algorithm 7, selects exemplar indices (with uniform probability for each class) to occupy each allocation. Within fillPtPop, a balanced representation from each non-converged class of the training data is enforced in lines

Algorithm 6 trainCmge(...) - Main training cycle of CMGE algoirthm.

Input: Assumes global read / write access to variables described in Table 3.1.
Output: Populations modified by a training cycle.

```
1: PP := fillPtPop( TD, NC )
2: TS := PA[1] ∪ ... PA[c] ∪ PP
3: evalArchives( LA, TS )
4: evalLearners( LP, TS, LA )
5: for 0 ... |LP| − 1 do
6:    p1 := select( LP )
7:    tries := 0
8:    p2 := ∅
9:    while class(p1) ≠ class(p2) ∧ tries < c do
10:      p2 := select( LP )
11:      tries := tries + 1
12:   end while
13:   if test( PXO ) then
14:      C := applyXover( p1, p2 )
15:   else
16:      C := {p1, p2}
17:      C := applyMutation( C, MR )
18:   end if
19:   evalLearners( C, TS, LA )
20:   replace( LP, C )
21: end for
22: epochs := epochs + 1
```

Algorithm 7 fillPtPop(TD, NC) - Selection of point population members.

Input: Training data TD, non-converged classes array NC.
Output: Subset S of training data indices having balanced representation from each class in NC.

```
1: S := {∅}
2: for i ∈ NC do
3:    TD_i := {t ∈ TD | class(t) = i}
4:    for 1 ... PP_size / |NC| do
5:       t := random( TD_i )
6:       S := S ∪ t
7:       TD_i := TD_i − t
8:    end for
9: end for
10: return S
```

Algorithm 8 select(LP) - Population selection operator

Input: Learner population LP upon which to apply selection.
Output: Returns a learner selected stochastically in proportion to fitness.

```
1: s := 0
2: sf := sum( LP.fitness )
3: r := random( {x ∈ ℝ+ | x < sf} )
4: for i := 0 ... |LP| − 1 do
5:    s := s + LP[i].fitness
6:    if r < s then
7:       return LP[i]
8:    end if
9: end for
```

Algorithm 9 applyXover($L1, L2$) - Crossover operator.

Input: Learners $L1$, $L2$ to apply crossover operator.
Output: Offspring $C1$, $C2$.

```
1: repeat
2:    if test( PCXO ) then
3:       pt := random( {x ∈ ℤ+ | x < CODONS ∧ L1.type[x] = L2.type[x] } )
4:    else
5:       pt := random( {x ∈ ℤ+ | x < CODONS} )
6:    end if
7:    for i := 0 ... CODONS − 1 do
8:       if i < pt then
9:          C1.codon[i] := L1.codon[i]
10:         C2.codon[i] := L2.codon[i]
11:      else
12:         C1.codon[i] := L2.codon[i]
13:         C2.codon[i] := L1.codon[i]
14:      end if
15:   end for
16: until geMap( C1.codon ) ∧ geMap( C2.codon )
17: return {C1, C2}
```

Algorithm 10 applyMutation(L, P) - Mutation operator.

Input: Learners L to apply mutation operator with a codon-wise probability of application P (%).
Output: Assumes L passed by reference.

```
1: for i := 0 ... |L| − 1 do
2:    repeat
3:       C := {c ∈ ℤ+ | c < CODONS}
4:       if test( PTSM ) then
5:          C := {c ∈ C | L[i].type[c] = −1} /* Terminal codons */
6:       end if
7:       for j := 0 ... |C| − 1 do
8:          if test( P ) then
9:             L[i].codon[j] := random( {x ∈ ℤ+ | x ≤ MAX_CODON_VAL} )
10:         end if
11:      end for
12:   until geMap( L[i].codon )
13: end for
```

Algorithm 11 evalArchives(LA, TS) - Evaluation of learner archives.

Input: Learner archives LA to be evaluated against the training set TS.
Output: Assumes LA passed by reference.

```
1: for i := 0 ... |LA| − 1 do
2:    caclGpOut( LA[i], TS )
3:    wrapGpOut( LA[i] )
4: end for
```

Algorithm 12 evalLearners(L, LA, TS) - Evaluation of learners on a training set.

Input: Learners L to be evaluated against the training set TS learner archives LA.
Output: Assumes L passed by reference.

```
1: for i := 0 ... |L| − 1 do
2:    caclGpOut( L[i], TS )
3:    assignLMF( L[i], TS )
4:    wrapGpOut( L[i] )
5:    evalObjectives( L[i], LA, TS )
6: end for
7: rankLearners( L )
```

Algorithm 13 calcGpOut(L, TS) - Mapping and evaluation of GE expression

Input: Learner L to evaluate raw GP outputs, Training set TS.
Output: Assumes L passed by reference.

1: $L.expression$:= geMap($L.codon$)
2: **for** $i := 0 \ldots |TS|$ **do**
3: $\quad L.gpOut[i]$:= eval($L.expression, TS[i]$)
4: **end for**

Algorithm 14 assignLMF(L, TS) - Assignment of local membership function

Input: Learner L to be assigned an LMF; Training set TS
Output: Assumes L passed by reference.

1: $[C, L.M]$:= potentialFn($L.gpOut$)
2: $L.\mu := TS[C[0]]$
3: $L.class$:= class($TS[L.\mu]$)
4: **for** $i := 0 \ldots |L.M| - 1$ **do**
5: $\quad s := s + (L.M[i] - L.\mu)^2$
6: **end for**
7: $L.\sigma := \sqrt{\frac{s}{|L.M|-1}}$

2 - 9. Line 3 selects all data from relevant classes, while lines 4 - 8 select from these patterns (without replacement, line 7) until the space allocated has been filled. Notably, the loop of line 4 ensures that the allocation for each non-converged class increases uniformly as classes converge. The function explicitly represents each class of the problem with equal exemplar counts in the point population assuming that there are enough points of each class in the training data to do so. The point population thus provides the basis for balancing the training set. Moreover, scalability of the training algorithm is achieved by the sampling property of the point population, as it explicitly limits the maximum number of learner evaluations required at the computationally costly inner loop of GP (line 19 of Algorithm 6).

3.7.2 The GE Learning Cycle

The MOGE framework induces GE classifier expressions through the multi-objective evolutionary cycle, Algorithm 6. The basic procedure for learner generation at each

Algorithm 15 wrapGpOut(L) - Application of local membership function

Input: Learner L.
Output: Assumes L passed by reference.

1: **for** $i = 0 \ldots |L.gpOut| - 1$ **do**
2: $\quad L.y[i] := \exp\left(-\frac{(L.gpOut[i]-L.\mu)^2}{2L.\sigma^2}\right)$
3: $\quad$ **if** $L.gpOut[i] \in [L.\mu \pm L.\sigma]$ **then**
4: $\quad\quad L.\hat{o}[i] := 1$
5: $\quad$ **else**
6: $\quad\quad L.\hat{o}[i] := 0$
7: $\quad$ **end if**
8: **end for**

Algorithm 16 evalObjectives(L, LA, TS) - Multi-objective evaluation of learner.

Input: Learner L to be evaluated according to Section 3.7; Learner archives LA and training set TS. Assumes access to string length function strlen.
Output: Assumes L passed by reference.

1: $L.Objective[0]$:= evalSSE(class($L.M$), $\{L.y[L.M]\}$)
2: $L.Objective[1]$:= countTruePositive($L.y$)
3: $L.Objective[2]$:= evalOverlap(L, LA, TS)
4: $L.Objective[3]$:= strlen($L.expression$)

Algorithm 17 evalSse(L, Y) - Evaluation of sum squared error.

Input: Labels L, outputs Y.
Output: e, the sum of squared errors.

1: $e := 0$
2: **for** $i := 0 \ldots |L| - 1$ **do**
3: $\quad e := e + (L[i] - Y[i])^2$
4: **end for**
5: **return** e

Algorithm 18 countTruePositive(L, TS) - Estimates true positives of a learner.

Input: Learner L to be evaluated with respect to TS.
Output: Returns the estimated count of true positives.

1: **for** $i := 0 \ldots |TS| - 1$ **do**
2: $\quad$ **if** $L.\hat{o}[i] = 1 \wedge \text{class}(TS[i]) = L.class$ **then**
3: $\quad\quad tp := tp + 1$
4: $\quad$ **end if**
5: **end for**
6: **return** tp

Algorithm 19 evalOverlap(L, LA, TS) - Evaluates learner 'overlap' with respect to Learner Archives LA.

Input: Learner L to be evaluated for overlap with respect to Learner Archives LA.
Output: Assumes L passed by reference.

```
1: L.overlap = 0
2: for i := 0 ... |TS| − 1 do
3:    if L.ô[i] = 1 ∧ class(TS[i]) = L.class then
4:       for j := 0 ... |LA[L.class]| − 1 do
5:          if LA[L.class][j].ô[i] = 1 ∧ class(TS[i]) = L.class then
6:             L.overlap := L.overlap + 1
7:          end if
8:       end for
9:    end if
10: end for
```

Algorithm 20 rankLearners(L) - Pareto ranking of learners.

Input: Learners L to be ranked by Pareto-ranking with ties.
Output: Assumes L passed by reference.

```
1: T := ∅
2: for i := 0 ... |L| − 1 do
3:    L[i].rank := 1
4: end for
5: for i := 0 ... |L| − 1 do
6:    for j := 0 ... |L| − 1 do
7:       r := aDomB( L[j], L[i] )
8:       if r = −1 then
9:          if i ≠ j ∧ {i, j} ∉ T then
10:            T := T ∪ {i, j}
11:            L[random( {i, j} )].rank + +
12:         end if
13:      else
14:         L[i].rank := L[i].rank + r /* Note: r is 1 or 0 */
15:      end if
16:   end for
17:   L[i].fitness := (|L| − L[i].rank) / |L|
18: end for
```

Algorithm 21 aDomB(A, B) - Pareto dominance comparison.

Input: Learners A, B to be compared over objective arrays. Without loss of generality, assumes minimization over all objectives; intended usage should be clear from context of calling function.

Output: 1 if A Pareto dominates B (i.e. $A \prec B$); -1 if A, B indifferent; 0 otherwise.

```
1: flag := 0
2: for i := 0 ... |A.Objective| − 1 do
3:    if A.Objecitve[i] ≤ B.Objective[i] then
4:       flag := flag + A.Objecitve[i] < B.Objective[i]
5:    else
6:       return 0
7:    end if
8: end for
9: if flag > 0 then
10:   return 1
11: else
12:   return -1 /* A and B are indifferent */
13: end if
```

Algorithm 22 replace(LP, C) - Population replacement operator.

Input: Learner population LP and children C to be inserted.

Output: Assumes L passed by reference.

```
1: for i := 0 ... |C| − 1 do
2:    [rank_min, i_min] := min( LP.rank )
3:    LP := LP − LP[i_min]
4:    LP := LP ∪ C[i]
5: end for
6: rankLearners( LP )
```

Algorithm 23 test(P) - Generic test for application of genetic operators

Input: Probability P (%) with which to apply the relevant test.

Output: 1 if test passes; 0 otherwise.

```
1: r := random( {x ∈ ℝ+ | x < 100} )
2: if r < P then
3:    return 1
4: else
5:    return 0
6: end if
```

Algorithm 24 random(S) - Random number selection.

Input: S, a set of values / objects.
Output: A random element of S.
1: $r :=$ rand(0,|S|-1)
2: **return** $S[r]$

iteration of step 5 in Algorithm 6 first involves the fitness proportionate stochastic selection of two parents, with an attempt to choose the second parent ($p2$) such that it has same class as the original, ($p1$). Should the selection function (select, Algorithm 8) fail to return a match after c attempts, the last individual returned is accepted as a default for $p2$ on line 10 of Algorithm 6 to avoid infinite loops. The select function provided in Algorithm 8 indicates a basic approach to the fitness proportionate selection scheme used in this work, which is further described in Section 1.4.4 of Chapter 2, i.e., is taken from standard GE practice.

The next step of the learning cycle involves the test for genetic variation operators. In the current framework, only one test (for crossover) is used (line 13, Algorithm 6) and has an associated probability of PXO, as defined in Table 4.5. If the test for crossover passes, then the crossover operator (applyXover, Algorithm 9) is applied to parents ($p1, p2$) to create the children ($C = \{c1, c2\}$). A failure of this test results in children being created as direct copies of parents, as indicated by the assignment on line 16. This assignment is then followed by a stochastic application of point-mutation (applyMutation, Algorithm 10). While the test function of Algorithm 23 is self-explanatory, the genetic operators are specially formulated for context sensitivity under GE and as such, employ the *type* information garnered from the geMap function of Algorithm 2 (describing 'canonical' GE as utilized in this work).

In the case of crossover (provided by applyXover, Algorithm 9), a test for context crossover takes place at line 2 with a probability of acceptance of PCXO (provided in Table 4.5). When the test passes, the codon crossover point is restricted to the stochastic selection of codons having matching (non-terminal) types on line 3. This ensures that the tail codons are used with the same non-terminal context in both individuals. A failure of the context crossover test results in the standard single point crossover procedure on line 5, Algorithm 9 [91]. In practice, the attempts at context

crossover obviously must be limited to avoid infinite loops when no legal crossover points exist. Moreover, the crossover algorithm repeats in the event that an illegal mapping occurs, e.g., a sequence of non-terminating rule selections or an expression exceeding the maximum string length. This is indicated by line 16 of Algorithm 9.

Point-mutation as implemented by the current framework (provided by apply-Mutation, Algorithm 10) may probabilistically employ a type-specific version of the mutation operator which is designed to minimize structural disruptions by strictly targeting terminal codons (this has been termed *terminal specific mutation* or TSM). The test for terminal specific mutation occurs at line 4 of Algorithm 10 with a probability of acceptance set to PTSM, supplied in Table 4.5. When the test passes, codons to be selected for mutation are constrained to those mapping to terminal rules only on line 5 (with terminal codons being assigned a value of -1 by the geMap function of Algorithm 2). Under the current framework, this implies that a terminal specific mutation can only alter a variable (or operator) with another variable (or operator) and therefore widespread structural changes to the structure of GE expressions are implicitly disallowed. In the event that the test for application of terminal specific mutation fails, the standard mutation points (any mapping codon) are permitted, as assigned in line 3 of Algorithm 10. The point (codon-wise) application of mutation takes place on line 9. It should be noted that the rate of actual point mutation (P) can be normalized to affect similar numbers of codons irrespective of the number of codons available to mutation, $|C|$. Similarly to crossover, the mutation algorithm repeats (line 12) in the event of an illegal mapping, e.g., a sequence of non-terminating rule selections or an expression exceeding the maximum string length.

Following the reproduction and application of genetic operators, the newly created offspring in C are evaluated under the multi-objective context by evalLearners (Algorithm 12). A detailed discussion of the multi-objective evaluation, ranking and fitness assignment is provided in Section 3.7.4.

The final step of the learning cycle involves replacement on line 20 of Algorithm 6. The replacement function (replace, Algorithm 22) follows the rule that children always replace the lowest ranked member of the learner population, LP. The replacement algorithm (line 2 Algorithm 22) does not preclude the replacement of the first child

by the second. Following the replacement step, the learner population is re-ranked by Algorithm 20 (rankLearners) and the next selection iteration begins.

3.7.3 Multi-objective Evaluation

Learner evaluation begins with the evalLearners function provided in Algorithm 12. This function provides the calculation of *gpOut* values (calling Algorithm 13, line 2), assignment of the Gaussian LMF, (calling Algorithm 14, line 3), application of the LMF (calling Algorithm 15, line 4) and finally evaluation of the learning objectives (calling Algorithm 16, line 5). Once all learners in L are evaluated, they are ranked in the last step of Algorithm 16 on line 7, with the call to rankLearners (Algorithm 20).

Within the context of this work, the interaction between a learner and a point is defined by evaluating the learner expression using the subset of point data and producing an output value (*gpOut*). The *gpOut* array for each individual is calculated by evaluating the learner's arithmetic GE expression against each exemplar of TS using the calcGpOut function provided in Algorithm 13. Line 1 calls the mapping function (Algorithm 2), which maps codons to an expression according to the global CFG, G. The expression is then evaluated against all members of the training set TS on lines 2 - 4. The expression evaluation is provided by an expression interpreter that is appropriate for the target grammar, G, written using the open source C tools for lexical analysis and compiler creation known as LEX and YACC[1], respectively. At this point we have established the mapping from the (typically multidimensional) input space to the one dimensional output space *gpOut*.

Next the local membership function of Equation 3.2 is assigned by assignLMF, Algorithm 14. The LMF parameters of μ and σ for a learner are established on line 1 by calling the clustering function known as the Potential Function (Algorithm 35), which is evaluated on the array of GP output values (*gpOut* points). Specifically, the Potential Function (Algorithm 35) returns the indices to the cluster centroids (C) and the cluster member indices (M). The Potential Function is described in detail in Section 3.11, however, we note that the choice of clustering function is not

[1]LEX and YACC are provided by http://dinosaur.compilertools.net/

significant as long as it is not necessary to supply the number of clusters *a priori*. Although generally considered to be a computationally expensive process, clustering is performed only over the balanced exemplar subset TS as dynamically identified by competitive coevolution. In doing so, sufficient in and out-of-class exemplars are provided for deriving the properties of the LMF. Moreover, the *gpOut* points correspond to the raw outputs of the learner expression and the clustering function is therefore a one-dimensional process applied with respect to each individual, providing:

1. μ: A single value on *gpOut*, defining the centroid of the largest / most dense cluster;

2. M: The set of cluster member points on *gpOut*. These correspond to the *gpOut* values near μ, as defined by a nearest neighbor allocation with respect to the two neighboring cluster centroids identified in the same *gpOut* distribution.

Algorithm 14, lines 2 and 3 assign the μ value (being the first, and therefore cluster centroid with highest density) and class (the label of the point associated with μ), respectively. Specifically, line 3 assigns the class label corresponding to the centroid μ to the learner, which is thereafter used to detect points of that class. Next (lines 4 - 7 of Algorithm 14), the member set M is used to estimate the LMF width as the standard deviation, σ, as indicated in Equation 3.1. A basic example of this is provided in Figure 3.8.

$$\sigma = \sqrt{\frac{1}{|M| - 1} \sum_{i=1}^{|M|} (M_i - \mu)^2} \tag{3.1}$$

The evalLearners algorithm resumes with the application of the LMF function, carried out by the call to wrapGpOut (Algorithm 15), on line 4 of Algorithm 12. The wrapGpOut function employs the Gaussian parameters, μ and σ to assign a membership (y value, Equation 3.2) to each element of *gpOut* on line 2 and provides an estimate of in vs. out-of-class (stored in the $\hat{o}$ array) according to Equation 3.3 on lines 4 and 6 respectively.

$$y\left(gpOut\right) = \exp\left(-\frac{(gpOut - \mu)^2}{2\sigma^2}\right) \tag{3.2}$$

$$\hat{o} = \begin{cases} 1 & \text{if } gpOut \in [\mu \pm \sigma] \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

The re-expression of the output represents a key departure from the standard GP approach to classification, where a GP classifier typically invokes a hard global switching function centered at zero to render the decision (i.e., if $gpOut \leq 0$ then return class 0, else return class 1), see Figure 2.3 (a). Here, the use of a local membership function (LMF) permits expressions to represent a subset of the data such that problem decomposition is facilitated and solutions take the form of several specialist classifiers rather than a single super individual. Moreover, at this stage no attempt is made to incorporate the concept of class membership. Instead the formation of LMFs with consistent class membership will be enforced through the multi-objective fitness evaluation, Section 3.7.4.

3.7.4 Objectives

The multi-objective evaluation is finally provided by the evalObjectives function of Algorithm 16 as called by line 5 of Algorithm 12. At this stage each learner has an output array (y) expressed in terms of a Gaussian LMF. The mean and variance of this LMF reflect the region of highest density relative to the mapping:

$$gpOut_i = f(x_i); i \in TS, \tag{3.4}$$

where $f(\cdot)$ is the mapping between multidimensional input x_i and single dimensional output $gpOut$ and i indexes training examples in the training set TS. The basic objective is now to incorporate class consistent properties onto the mapping of (3.4) and therefore the points associated with the LMF.

Central to establishing the objectives are the concepts of: 1) error, relative to exemplar class and degree of LMF membership; 2) in-class membership count, where this is defined with respect to the region defined by the LMF; 3) overlap minimization where in-class exemplars are discouraged from being a member of more than one LMF; 4) solution parsimony, in terms of expression string length. The definitions of these objectives are summarized as follows:

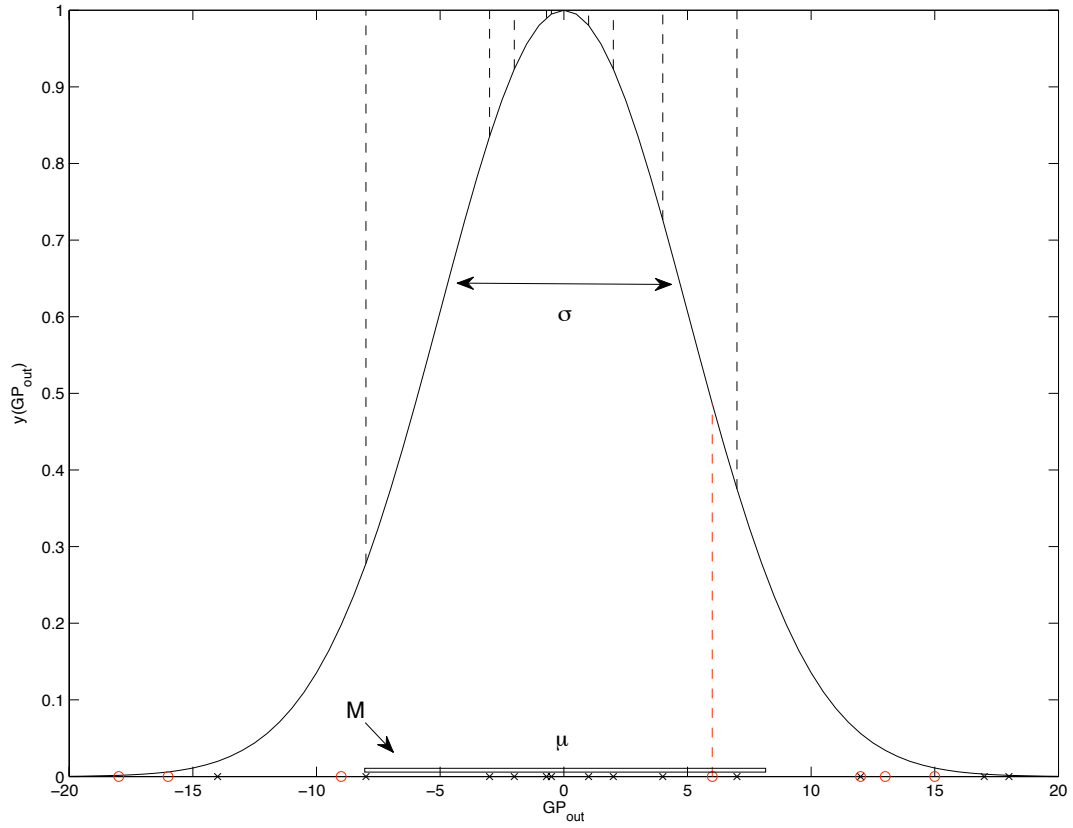


Figure 3.8: CMGE (Gaussian) LMF and associated error evaluation on cluster members M; 'x' points on GP_{out} indicate mappings corresponding to in-class exemplars and 'o' points indicate out-of-class exemplars. Associated error terms are indicated by dashed lines (see Equation 3.5).

1. **Minimize the sum of squared error (SSE):** This objective explicitly enforces cluster class-consistency by evaluating classification performance over the cluster member points and rewarding true positive classifications while discouraging the occurrence of false positives (an individual mistakenly labeling a pattern of class 0 as class 1). As provided by the call to Algorithm 17, evalSse, from line 1 of Algorithm 16, the SSE for an individual is calculated over M, the set of cluster member points (returned by the Potential Function, Algorithm 35):

$$\text{SSE} = \sum_{i=1}^{|M|} (label_i - y\,(gpOut_i))^2, \tag{3.5}$$

with y as defined in (3.2) and *label* taking on binary values: 1 when the current pattern label is the same as the class as that of the current learner (as determined by the clustering) and 0 otherwise. We emphasize that SSE is only estimated over the subset of exemplars associated with the LMF, not all exemplars within the point population (Algorithm 16, line 1). That is to say, as long as the LMF is consistent in its classification we are not interested in the performance on exemplars outside of the LMF neighborhood. Such a bias naturally encourages problem decomposition or specialization of learner behavior.

2. **Maximize in-class patterns correctly classified:** This objective is designed to encourage survival of individuals that correctly map many patterns densely in *gpOut*, while discouraging the case of single point coverage by classifiers. This objective balances the specialization effect of objective 1 and is provided by the call to Algorithm 18, countTruePositive, from line 2 of Algorithm 16.

3. **Minimize pattern overlap:** This objective is intended to discourage intersection in the sets of patterns that are correctly classified between learner archives and the current learner population. As given in Algorithm 19 (called on line 3 of Algorithm 16), the overlap value for an individual is defined as a count (i.e., sum) of number of times that each exemplar that is correctly classified is also correctly classified by other members of the relevant learner archive. Each case

of overlap (an exemplar satisfying the conditions of lines 3 and 5 of Algorithm 19), adds 1 to the learner's overlap count on line 6. The overlap count is calculated over the training set TS (line 2) against all members of the relevant learner archive $LA[L.class]$ (line 4). This approach to overlap count makes a significant improvement relative to an earlier version of the MOGE classifier of [82].

4. **Minimize expression length:** Based on results obtained in [93] and [25], this objective imposes parsimony in learner expressions. The expression length is defined as the string length of the unsimplified learner expression and is provided by the call to the strlen function on line 4 of Algorithm 16).

In summary, the SSE objective captures the quality of the exemplars mapped to the same LMF; whereas in-class exemplar count naturally discourages degenerate behavior. Pattern overlap is a global metric in that it is estimated with respect to exemplar 'coverage' of other learners in the archive. On their own these objectives would not necessarily lead to useful classifiers (e.g., minimizing the SSE can be achieved without including any in-class exemplars). Taken together, however, they fully establish properties for a 'good' classifier whilst also encouraging problem decomposition within exemplars associated with the same class. The fourth objective, parsimony, naturally assumes that classification performed on the basis of multiple (non linear) mappings is more appropriate than relying on a single mapping, especially with respect to solution transparency.

Lastly, an overall rank (and therefore fitness) of learners is established by the call to rankLearners (Algorithm 20) on line 7 of the learner evaluation, Algorithm 12. Learners are ranked using the Pareto-ranking with ties algorithm, described in Section 3.7.5.

3.7.5 Pareto Ranking and Fitness Assignment.

The method of ranking with ties, Algorithm 20 (called on line 7 of Algorithm 12), is employed to determine fitness of members of the learner population [66]. This approach was originally used in MOGA algorithm of Fonseca and Fleming [37] and

begins by assigning a rank of 1 to all learners L on lines 2-4. Next a double loop over the learners is used to compare each learner to all others in terms of the dominance criteria on lines 5-7. The rank of an individual is defined by the number of individuals by which it is Pareto dominated plus one, where Pareto dominance is described in Section 2.3.1 and implemented according to Algorithm 21, aDomB. Rank increments are provided by line 14 of Algorithm 20 in the event of domination by another learner (i.e., a return value of 1 is assigned to r on line 7). All non-dominated solutions therefore retain the original rank of 1, and in the event of a tie (i.e., two learners having the same value in all objectives), a -1 is returned by Algorithm 21 on line 7 of Algorithm 20, and one of the ranks is randomly increased by one on line 11. The fitness of an individual is finally assigned in direct proportion to Pareto rank on line 17 of Algorithm 20.

3.8 Point and Learner Archive Entry

Following each multi-objective learning cycle (step 4 of Algorithm 3) corresponding to the first stage, steps 1-3 of the high level Figure 3.7), an archiving process begins, representing the second logical stage of the CMGE algorithm (steps 4-6 of Figure 3.7). The archive evaluations (ipcaEvaluation, Algorithm 25 called on line 4 of Algorithm 3) are driven by the notion of providing *distinctions* between learners [34]. The particular model that we follow takes as a basis de Jong's IPCA algorithm [23]. The concept of distinctions was shown to specifically address the coevolutionary problem of disengagement [24], where the point population dominate the learner population resulting in a loss of training gradient. This can occur when points are rewarded for explicitly defeating the learners rather than distinguishing between them.

The archiving approach employed here differs from IPCA [23] and Lemczyk's binary adaptation [70] [71] in two respects. First, multiple independent archives are maintained concurrently, each corresponding to a different class of the problem. This is necessary in order to ensure interactions maintain a training gradient that is relevant with respect to each class. Within each point archive, we enforce a 50-50 balance between {in, out}-of-class points on the archive contents (where the notion of an 'in' or 'out-of-class' point is obviously with respect to the class of the archive).

Secondly, the definition of an outcome has been reformulated to take on real values as opposed to binary in the outcome evaluation of Algorithm 26 (calcOutcomes). An outcome is the result of an interaction between a learner and a point. The set of outcomes defined for each learner with respect to its corresponding archive provides the basis for which archive entry is assessed in Algorithm 25 (ipcaEvaluation). In this work, outcomes are evaluated with respect the class of the learner and take on real values in the range (0,1] based on the the learner's membership y as defined in (3.2). Specifically, Algorithm 26, lines 6 and 8 define a learner's outcome for a given point as:

$$outcome = \begin{cases} y & \text{if point is in-class} \\ 1 - y & \text{otherwise} \end{cases} \tag{3.6}$$

3.8.1 Archive Entry Criteria: Distinctions and Usefulness

Both point and learner archives are driven by the notion of *distinctions* [34]. Consistent with IPCA, a point is said to provide a distinction between learners (i.e., it is considered to be a *useful* test) if the outcomes of a learner that was previously Pareto dominated by the learner archive become Pareto equivalent to the learner archive with the addition of the point. In this situation, the learner in question is said to become *useful* with the addition of the point to the point archive.

On each call to the high level archive evaluation, Algorithm 25 (ipcaEvaluation), IPCA specifies that all useful points and learners are committed to the archives. This will be relevant for CMGE in the non-converged classes (those in NC), therefore the learner archive outcomes in relation to the point archives are calculated at the outset (lines 2-4) for the non-converged classes only. Finding useful points thus first requires identification of all useful and non-useful learners (lines 5-12). On line 7 the usefulness of learners is determined by Algorithm 27, isUseful. Usefulness of a learner L with respect to a set of learners LS is established in the CMGE context if the learner is non-dominated and unique in terms of the real-valued outcome vectors. This is satisfied when the aDomB function (of Algorithm 21) returns 0 for every learner in LS evaluated against L, on line 2.

Next the useful points are identified in Algorithm 25 as those promoting some

non-useful learner to useful (line 17). Learners thus promoted (becoming useful with the relevant point addition) are retained on line 16 to be later evaluated and potentially committed to the learner archives. Line 24 passes the useful points UP and useful learners UL along with the current archives and the number of classes c to the updateArchives function (Algorithm 28) for final learner evaluations and commitment of learners and points.

3.8.2 Class-specific Archive Update Rules

CMGE's archive update function (updateArchives, Algorithm 28) requires class-specific commitment of points and learners in order to maintain learners and points that have gradient relevance with respect to their classes. Specifically, points that have previously been deemed useful ($j \in UP$) with respect to a learner (of class i) and the class i learner archive $LA[i]$, are committed to the class i point archive $PA[c]$ on lines 1-6.

Next, lines 7-10 of Algorithm 28 re-evaluate the previously deemed useful learners UL against their respective (newly updated) point archives. Those again deemed useful (line 10) are committed to the learner archive LA matching their class lc (line 12 or 14 for CMGE1 and CMGE2, respectively). Finally lines 16-22 check for the case of an undefeated learner. When a learner has no outcome less than 1 (line 16), it is undefeated and a point should be inserted into the undefeated learner's corresponding point archive in order to guide the next cycle of learning towards improvement. To do so, the outcomes of the undefeated learner are calculated relative to the point archive (line 17) and the point resulting in the minimum outcome (line 18) is added to the point archive (line 20).

3.8.3 Archive Insertions and Pruning

In order to maintain an upper bound on the archives, pruning may be necessary prior to insertion of the new learner or point member. A maximum size LA_{size}, PA_{size} is therefore imposed on the learner and point archives, respectively, in order to sustain

Algorithm 25 ipcaEvaluation(...) - IPCA archive entry criteria evaluations.

Input: Assumes global read / write access to variables described in Table 3.1.
Output: Modified learner, point archives.

```
 1: UL := {∅}; UL̄ := {∅}; UP := {∅}
 2: for i ∈ NC do
 3:    calcOutcomes( LA[i], PA[i] )
 4: end for
 5: for i := 0 ... |LP| − 1 do
 6:    calcOutcomes( LP[i], PA[LP[i].class] )
 7:    if ! isUseful( LP[i], LA[LP[i].class] ) then
 8:       UL̄ := UL̄ ∪ LP[i]
 9:    else
10:       UL := UL ∪ LP[i]
11:    end if
12: end for
13: for i := 0 ... |UL̄| − 1 do
14:    lc := UL̄[i].class
15:    for j := 0 ... |PP| − 1 do
16:       calcOutcomes( LA[lc], PA[lc] ∪ PP[j] )
17:       calcOutcomes( UL̄[i], PA[lc] ∪ PP[j] )
18:       if isUseful( UL̄[i], LA[lc] ) then
19:          UL := UL ∪ UL̄[i]
20:          UP[lc] := UP[lc] ∪ PP[j]
21:       end if
22:    end for
23: end for
24: updateArchives( UL, LA, UP, PA, c )
```

Algorithm 26 calcOutcomes(L, P) - IPCA outcome vector evaluation.

Input: Learners L to find real-valued outcomes on points P.
Output: Assumes L passed by reference.

```
 1: for i := 0 ... |L| − 1 do
 2:    calcGpOut( L[i], P )
 3:    wrapGpOut( L[i] )
 4:    for j := 0 ... |P| − 1 do
 5:       if L[i].class = class( P[j] ) then
 6:          L[i].outcome[j] := L[i].y[j]
 7:       else
 8:          L[i].outcome[j] := 1 − L[i].y[j]
 9:       end if
10:    end for
11: end for
```

Algorithm 27 isUseful(L, LS) - IPCA 'useful' evaluation.

Input: Learner L outcome array is evaluated against the outcome arrays of the set of learners LS to determine usefulness (i.e. non-dominated and unique).

Output: 1 if useful; 0 otherwise.

```
1: for i := 0 ... |LS| − 1 do
2:    if aDomB( LS[i], L ) = ± 1 then
3:       return 0
4:    end if
5: end for
6: return 1
```

Algorithm 28 updateArchives(UL, LA, UP, PA, PP, c) - IPCA archive updates.

Input: Useful Learners UL, learner archives LA, useful points UP, point archives PA, point population PP, number of classes c.

Output: Assumes LA, PA passed by reference.

```
1: for i := 0 ... c − 1 do
2:    for j := 0 ... |UP[i]| − 1 do
3:       archivePoint( UP[i][j], PA[i] )
4:    end for
5:    calcOutcomes( LA[i], PA[i] )
6: end for
7: for i := 0 ... |UL| − 1 do
8:    lc := UL[i].class
9:    calcOutcomes( UL[i], PA[lc] )
10:   if isUseful( UL[i], LA[lc] ) then
11:      if CMGE1 then
12:         archiveLearner( UL[i], LA[lc] ) /* CMGE1: Greedy prune */
13:      else
14:         archiveLearnerTs( UL[i], LA[lc], PA[lc] ) /* CMGE2: Two stage prune */
15:      end if
16:      if ∄ o ∈ UL[i].outcome | o < 1 then
17:         calcOutcomes( UL[i], PP )
18:         [o_min, i_min] := min( UL[i].outcome )
19:         if o_min < 1 then
20:            archivePoint( PP[i_min], PA[lc] )
21:         end if
22:      end if
23:   end if
24: end for
```

Algorithm 29 archiveLearner(L, LA) - Greedy learner archive insertion.

Input: Learner L to be archived to learner archive LA.
Output: Assumes LA passed by reference.

if $|LA| < LA_{size}$ **then**
  $LA := LA \cup L$
else
  $i_{min} := 0$
  $s_{min} := LA_{size}$
  for $i := 0 \ldots |LA| - 1$ **do**
    if $s :=$ sum($LA[i].outcome$) $< s_{min}$ **then**
      $s_{min} := s$
      $i_{min} := i$
    end if
  end for
  $LA[i_{min}] := L$
end if

Algorithm 30 archiveLearnerTs(L, LA, PA) - Two-stage learner archive insertion.

Input: Learner L to be archived to archive LA; corresponding point archive PA.
Output: Assumes LA passed by reference.

if $|LA| < LA_{size}$ **then**
  $LA := LA \cup L$
else
  $i_{min} := 0$
  $s_{min} := LA_{size}$
  $L_{outs} := \{L.outcome[p_{out}] \mid p_{out} \in PA \wedge \text{class}(PA[p_{out}]) \neq L.class\}$
  for $i := 0 \ldots |LA| - 1$ **do**
    $LA_{outs} := \{LA[i].outcome[p_{out}] \mid p_{out} \in PA \wedge \text{class}(PA[p_{out}]) \neq L.class\}$
    if $s :=$ sum(LA_{outs}) $< s_{min}$ **then**
      $s_{min} := s$
      $i_{min} := i$
    end if
  end for
  if $s_{min} < \text{sum}(L_{outs})$ **then**
    $LA[i_{min}] := L$
  else
    archiveLearner(L, LA)
  end if
end if

Algorithm 31 archivePoint(P, PA, c_{PA}) - IPCA point archive insertion.

Input: Point P to be archived to point archive PA (of class c_{PA}).
Output: Assumes PA passed by reference.

```
1: if class( P ) = c_PA then
2:    PA_c := {p ∈ PA | class(p) = c_PA}
3: else
4:    PA_c := {p ∈ PA | class(p) ≠ c_PA}
5: end if
6: if |PA_c| < PA_size/2 then
7:    PA := PA ∪ P
8: else
9:    [d_min, i_min] := min( dist( P, PA_c ) )
10:   PA_c[i_min] := P
11:   PA := (PA ∩ PA_c) ∪ P
12: end if
```

computational and resource efficiency in the training algorithm. Efficiency is therefore achieved at the potential expense of accuracy of learner evaluation.

In the case of point archiving (Algorithm 31, archivePoint) the point is simply added to the archive so long as the archive has not reached the 50% in or 50% out-of-class capacity ($\frac{PA_{size}}{2}$), for an in or out-of-class point, respectively (lines 6-8). When the capacity has been reached, the point (of the same class) having the minimum Euclidean distance to the incoming point is removed (pruned) in favor of the new point (lines 9-12). The distance is calculated over the pattern attributes (feature space) associated with each point [70]. While a point is lost from the archive through this process, the assumption is that the point inserted into the archive will provide an alternative test that maintains the previous distinctions while establishing a basis for further learning.

When a learner archive has reached capacity (LA_{size}), a member must be chosen for replacement. In the case of the learner archive, pruning risks correctly identifying a complete set of expressions that are able to decompose the classification problem, while in the case of the point archive pruning may introduce errors in identifying training objectives and cause cycles of forgetting in the learning process. The two variants of the CMGE algorithm benchmarked in this thesis are distinguished by their approaches to learner pruning as described in the following sections.

CMGE1

The CMGE1 is distinguished by the learner archiving algorithm (Algorithm 29, archiveLearner). Algorithm 29 simply adds the new learner to the archive when the capacity has not been reached (lines 1-3); otherwise the algorithm proceeds by pruning the learner archive member having the lowest sum of real-valued outcomes on lines 6-10 (against its class-appropriate archive), which is then replaced by the incoming learner (line 12). This pruning architecture is meant to encourage strong, accurate decisions among learner archive members. Moreover the pruning policy does not discriminate between the types of errors made by the classifiers (i.e., error that would contribute toward false positives vs. false negatives); when all else is the same, pruning therefore favors the learner archive members making strongly decisive classifications over sheer numbers of correct classifications as predicted by, for example, the binary ($\hat{o}$) outputs. This policy was designed to encourage survival of potentially cooperative team members by pruning individuals that respond weakly to archive points and may therefore cause conflicts in team-based classifications post-training.

CMGE2

The learner archiving algorithm distinguishing CMGE2 from CMGE1 (Algorithm 30, archiveLearnerTs) employs a two part greedy learner pruning policy as follows: the learner archive member having the lowest sum of real-valued outcomes (against its class-appropriate archive) across *out-of-class points* (identified by lines 7-13) is replaced by the incoming learner if the new learner represents an improvement in this respect (lines 14-15). Otherwise, the archive member having the lowest sum of outcomes is replaced as in CMGE1 (lines 16-18). This policy therefore discriminates against false positive error in the archive members by preferring to prune according to error type; however, when the incoming learner has a lower sum of out-of-class outcomes, the replacement policy reverts to the default policy of CMGE1, described above.

This pruning policy aims first to reduce the potential for false positive errors while encouraging strong, class-consistent decisions in learner archive members regardless of the number of total correct classifications. This policy was designed to explicitly

reduce the occurrence of false positives and thereby improve overall classification performance. Moreover, the policy also encourages the survival of potentially cooperative team members by pruning individuals that respond weakly to archive points and may therefore cause conflicts in team-based classifications post-training.

3.9 Early Stopping Criteria

To identify a converged state among learner population members (step 5 of Algorithm 3) we employ the convenient stopping criteria identification method of Pareto-rank histograms, introduced under a GA context by Kumar and Rockett [66]. CMGE employs rank histograms, which are generated from the ratio of the number of learners at each rank in the learner population between the current and previous epochs, are employed to identify class-wise early stopping in Algorithm 32, stopCmge.

Lines 1 and 2 indicate that separate rank histograms are generated (by the call to Algorithm 33, rankHist) for each non-converged class in NC in the learner population LP, such that at most c histograms are constructed for a c class problem.

Class-wise rank histograms are generated by Algorithm 33, rankHist based on [66], are generated with respect to learner populations of successive epochs (LP_t, LP_{t-1}) by first combining learners of the same class from both populations (into LP_c) and re-ranking all learners on lines 1 and 2, respectively. Each rank histogram entry (in R) is a ratio of the number of learners of class c in the current population LP_t to the number of learners having the same rank in LP_c as determined in the loop of lines 3-7.

Algorithm 32, stopCmge, continues on line 3, where a match (defined as a distance less than MIN_DIFF, Table 4.5) between learner population class rank histograms of successive epochs (RH, RH_{t-1}) indicates that further progress is unlikely on the class [66], and the class is therefore removed from the non-converged array NC on line 4[2]. If more classes remain (skipping lines 6-8), the learners associated with the

[2]For pragmatic reasons, step 3 of Algorithm 32 also requires that the number of learners corresponding to the histogram class under consideration must be beyond a minimum threshold

Algorithm 32 stopCmge(...) - Evaluation of early stopping criteria.

Input: Assumes global read / write access to variables described in Table 3.1.
Output: 1 if a stopping criterion has been reached; 0 otherwise.

```
1: for i := 0 ... |NC| − 1 do
2:    RH[NC[i]] := rankHist( LP, LP_{t−1}, NC[i] )
3:    if dist( RH[NC[i]], RH_{t−1}[NC[i]] ) < MIN_DIFF then
4:       NC := NC − NC[i]
5:       if |NC| = 0 ∨ epochs = MAX_EPOCHS then
6:          Stop criteria = 1
7:          return 1
8:       end if
9:       initLearners( {l ∈ LP | class(l) = NC[i]} )
10:   else
11:      RH_{t−1}[NC[i]] := RH[NC[i]]
12:   end if
13: end for
14: LP_{t−1} := LP
15: return 0
```

Algorithm 33 rankHist(LP_t, LP_{t-1}, c) - Rank histogram calculation.

Input: Current learner population LP_t, previous learner population LP_{t-1} and class c over which to calculate rank histogram.
Output: R, rank histogram.

```
1: LP_c := {l ∈ LP_t ∪ LP_{t−1} | class(l) = c}
2: rankLearners( LP_c )
3: for i := 1 ... LP_size do
4:    R_t[i] := |{l ∈ LP_t | l.rank = i ∧ class(l) = c}|
5:    R_c[i] := |{l ∈ LP_c | l.rank = i}|
6:    R[i] := R_t[i] / R_c[i]
7: end for
8: return R
```

newly converged class are then re-distributed (to non-converged classes) by the call to initLearners (Algorithm 5) on line 9. Moreover, lines 5-8 stop the main loop of cmgeMain (lines 2-6 of Algorithm 3) when the NC array is empty or if the training cycle counter *epochs* exceeds MAX_EPOCHS (Table 4.5).

3.10 Post-training: Assembly of Classifiers

Algorithm 34 buildSolution(LA, TD) - Returns final solutions with weights.

Input: Learner Archives LA and training data TD.
Output: Final solution (weighted learners) S.

1: $S := LA[1] \cup \ldots \cup LA[c]$
2: **for** $l := 0 \ldots |S| - 1$ **do**
3: **for** $i := 1 \ldots c$ **do**
4: $\hat{n}[i] := |\{pt \in TD \mid S[l].\hat{o}[pt] = 1 \wedge \text{class}(pt) = i\}|$
5: $t[i] := |\{pt \in TD \mid \text{class}(pt) = i\}|$
6: **end for**
7: $\hat{N} := \text{sum}(\hat{n})$
8: **for** $i := 1 \ldots c$ **do**
9: $F[i] := \left(\frac{\hat{n}[i]}{t[i]}\right)\left(\frac{\hat{n}[i]}{\hat{N}}\right)$ /* 'Favorability' for class i [102] */
10: **end for**
11: $weight := \frac{\max(F)}{\text{sum}(F)}$
12: $S[l].w := weight$
13: **end for**
14: **return** S

The main CMGE loop (lines 2-6 of Algorithm 3) is concluded when a stopping criterion is reached as described in Section 3.9, above. Post-training, the learner archives are finally merged on line 7 (Algorithm 3) to form a solution set S with the call to buildSolution (Algorithm 34), which is responsible for processing the learner archives to identify the degree to which each of the individuals will participate in making classifications. Such a model assumes that classifiers are deployed in parallel, thus supporting multi-class as well as single label applications. In this work, we implement a basic winner-take-all scheme where the individual's y values (as defined in (3.2)) are weighted with a confidence, w. The class of the individual having the

(CONV_MIN_POP) and that the current cycle of learning *epoch* (see Algorithm 6, trainCmge) must be beyond a minimum fraction (CONV_FRAC) of MAX_EPOCHS. For parameter specifications, see Table 4.5.

highest confidence weighted membership value ($w \cdot y$) is predicted by the classifier for a given input pattern. The confidence weighting ($w \in [0, 1]$) for each individual in the solution set is calculated and assigned on lines 11-12 in terms of its favorability set as:

$$w = \frac{max\,(F)}{\sum F}. \tag{3.7}$$

The elements of the favorability set F (established on lines 8-10) for an individual are defined over the classes ($i = 1 \ldots c$) of the entire set of training data TD as [102]:

$$F[i] := \left(\frac{\hat{n}[i]}{t[i]}\right)\left(\frac{\hat{n}[i]}{\hat{N}}\right) \tag{3.8}$$

where

$$\hat{n}[i] = |\{pt \in TD : \hat{o} = 1 \wedge class(pt) = i\}|, \tag{3.9}$$

$$t[i] = |\{pt \in TD : class(pt) = i\}|, \tag{3.10}$$

and

$$\hat{N} = \sum_{i=1}^{c} \hat{n}_i. \tag{3.11}$$

are calculated over the loop of lines 3-6, with $\hat{N}$ being assigned on line 7. The favorability for a class c, $F[c]$, is interpreted as the joint probability of a pattern of class c being classified by learner l and an exemplar classified by l being of class c [102]. The weight calculation of line 11 therefore normalizes the maximum probability by the sum of all (class-wise) probabilities, bringing the final weight w into the range [0,1].

3.11 The Potential Function

The Potential Function algorithm is an iterative process for defining cluster centers and hard cluster memberships given an array of points in n dimensions (for the current application this is 1 dimension) [18]. The algorithm does not make any assumptions regarding the number of clusters present in the data, but is based on a Gaussian

Algorithm 35 potentialFn(X) - Clustering by the Potential Function.

Input: An array of real valued points X
Output: An array of cluster centers C

1: $C := \{\varnothing\}$
2: **for** $i := 0 \ldots |X| - 1$ **do**
3: $\quad P[i] := 0$
4: $\quad$ **for** $j := 0 \ldots |X| - 1$ **do**
5: $\quad\quad P[i] := P[i] + \exp(-\alpha \|X[i] - X[j]\|^2)$
6: $\quad$ **end for**
7: **end for**
8: $P^*_{max} := \max(\ P\ )$
9: **loop**
10: $\quad [P_{max}, i_{max}] := \max(\ P\ )$
11: $\quad$ **if** $P_{max} > \gamma_{upper} \cdot (P^*_{max})$ **then**
12: $\quad\quad C := C \cup i_{max}$
13: $\quad\quad$ **for** $i := 0 \ldots |X| - 1$ **do**
14: $\quad\quad\quad P[i] := P[i] - P_{max} \cdot \exp(-\beta \|X[i] - X[i_{max}]\|^2)$
15: $\quad\quad$ **end for**
16: $\quad$ **end if**
17: $\quad X := X - X[i_{max}]$
18: $\quad$ **if** $P_{max} < \gamma_{lower} \cdot (P^*_{max}) \vee |X| = 0$ **then**
19: $\quad\quad$ **return** C
20: $\quad$ **end if**
21: **end loop**

Algorithm 36 dist(P, Q) - Euclidean distance calculation.

Input: Numeric arrays (points) P and Q having same dimensions.
Output: Euclidean distance d between P and Q.

1: $d := 0$
2: **for** $i := 0 \ldots |P| - 1$ **do**
3: $\quad d := d + (P[i] - Q[i])^2$
4: **end for**
5: **return** $\sqrt{d}$

kernel and as such requires *a priori* declaration of two radii parameters, α and β. Beyond requiring that the number of clusters not be specified *a priori*, any clustering algorithm would be appropriate.

The algorithm (Algorithm 35, potentialFn) begins by identifying each point's candidate *potential* with respect to all other points, using a suitable distance metric (Algorithm 35, lines 2 to 7). The distance metric is referred to as the *Potential Function* where α provides a means to influence the granularity of clusters on line 5. Points having the greatest degree of similarity to the current point, $X[i]$, contribute the most to the corresponding potential $P[i]$. Points having many neighbours in near proximity will therefore be assigned the greatest potentials.

Next the value of the highest total initial potential P^*_{max} is identified in Algorithm 35, line 8. The point (i_{max}) having the highest current potential (P_{max}) is then tested with respect to $\gamma_{upper} \cdot P^*_{max}$ to determine the creation of a new cluster center on lines 10 to 11, where $\gamma_{upper} \in (0, 1]$ is chosen as a fraction of the initial potential required to constitute a new cluster.

If the current value of P_{max} is sufficiently large, the assignment to the set of cluster centers C on line 12 then proceeds and this point's influence on the remaining points is removed in lines 13 to 15, where β ($> \alpha$) is the radius associated with the Potential decay process and thus each cluster member's potential is reduced by an amount proportional to its distance from the current maximum potential P_{max}.

The current i_{max} point is removed from further consideration on line 17, and the entire process iterates until a successful test for the end condition on line 18 triggers the return of the cluster set C, where $\gamma_{lower} \in [0, 1]$ is chosen as a fraction of the initial potential required to stop further consideration of new clusters.

The assignment of points $X[i]$ to clusters is determined by the point $X[i_{max}]$ corresponding to the greatest reduction in potential $P[i]$ during the decay loop.

Parameter values for all experiments requiring the Potential Function are provided in Table 4.5.

3.12 Summary

Having reviewed the related literature in chapter 2, we present a holistic approach for dealing with the factors central to reducing the computational overheads of GP based classification under binary and multi-class domains. Relative to the five factors of Equation (1.1) we identify the following properties in our approach:

Number of GP runs : Naturally, by assuming a stochastic learning algorithm, we expect to perform repeated trials for different initializations of the model populations and stochastic decision makers. However, we might speculate that by being able to decompose the problem into smaller / easier subproblems, that the degree of variance in the solutions is lower than when a single 'super' classifier is sought.

Number of classes : The proposed model evolves all classifiers from a single population, thus avoiding the need to perform as many sets of GP runs as classes.

Generation limit : The use of a model for dynamically re-directing learners implicitly provides a mechanism for detecting early stopping conditions. Moreover, the approach is based on an analysis of Pareto archive behavior, rather than 'error', and as such avoids the problem of setting problem dependent convergence thresholds.

Population size : This is indirectly adapted through the class wise stopping criterion, with classes converging early resulting in their search resource being made available for the remaining classes. As such we gain a mechanism for resizing the population class wise, under a constant population limit.

Data set exemplar count : Adoption of a competitive coevolutionary model enables us to efficiently sub sample the original training data, independently of the overall exemplar count.

As such, the proposed algorithm has an evaluation count of the following form:

$$Evals = I \times P \times SS \times ES \tag{3.12}$$

where I is the number of initializations, P is the population size, SS is the subset size, and ES is the worst case epoch limit ($ES <=$ MAX_EPOCHS).

Relative to the original expression, Equation 1.2, we have dropped the requirement to iterate over each class, and reduced $|TD|$ to SS and G need not rely on an a priori, possibly conservative estimate, of the generational limit. In addition, we have incorporated a parsimony bias, care of the multi-objective fitness function, thus further reducing the cost of the inner loop relative to that of canonical GP.

Central to achieving these 'speedups' was adopting a mechanism for describing class membership in terms of a local membership function, without having to resort to the heuristic partitioning of the *gpOut* axis. Evolutionary multi-objective optimization is then used to guide the evolution of desirable subsets of non-overlapping exemplars to a class consistent membership profile. A natural consequence of this mechanism is that we how have a model for problem decomposition, whereas previous approaches at best, produced solutions in the form of one individual per class [93], [121]. The integration of EMO with competitive coevolution provides the basis for focusing the search process on the most important goals, whilst doing so with a minimal computational footprint. Finally, we explicitly make use of previous research on appropriate biases for sampling the larger data set by assuming a class balance heuristic. This is particularly important as competitive coevolution is not able to direct the sampling of test points until useful test points have been discovered.

3.13 Computational Complexity Analysis

The time complexity of each algorithm provided in this chapter will now be analyzed with brief discussions appearing in Table 3.2. The time complexities of the high level functions are addressed in more detail in the sections below.

3.13.1 Extended Complexity Discussion

For the high level functions discussed in the next sections, the following simplifying assumptions are proposed:

1. Point dimension is small, fixed for a problem;

2. $S = LA_{size} \simeq PA_{size} \simeq PP_{size} \simeq LP_{size}$;

3. Number of classes, $c << S$.

cmgeMain

The time complexity of the main function is dominated by the while loop, which iterates until stop criteria have been met. At worst, this involves MAX_EPOCHS iterations over high level functions: trainCmge ($\mathcal{O}(S^2 + S \cdot |TS|^2)$), ipcaEvaluation ($\mathcal{O}(S^4)$) and stopCmge ($\mathcal{O}(S^2)$). Finally, the post training call to buildSolutions is $\mathcal{O}(c \cdot LA_{size} \cdot |TD|)$. These complexities are additive and the overall complexity is $\mathcal{O}(\text{MAX\_EPOCHS} \cdot (S^4 + S \cdot |TS|^2) + c \cdot LA_{size} \cdot |TD|)$, assuming that the simplifying assumptions are permitted. Each high level function is discussed in detail below. Derivations of time complexities for low level functions are provided in Table 3.2.

trainCmge

The first section of this function requires a call to fillPtPop ($\mathcal{O}(|PP|)$) and then a call to evalArchives and evalLearners. This costs on the order of $\mathcal{O}(|LP| \cdot |TS|^2)$, which dominates the complexity of this first section.

The second section begins with the for loop on line 5 which is over $|LP|$ and iteratively calls the following functions of significance: select, replace ($\mathcal{O}(|LP|)$), each; applyXover, applyMutation $\mathcal{O}(\text{CODONS})$, each; and evalLearners ($\mathcal{O}(|L| \cdot |TS|^2 + |L|^2)$) which in this context can be reduced to $\mathcal{O}(|TS|^2)$, since $|L| = |C| = 2$. Ignoring the $\mathcal{O}(\text{CODONS})$ term, this leaves $\mathcal{O}(|LP|^2 + |LP| \cdot |TS|^2)$. When additively combined with the cost of the first section provides $\mathcal{O}(|PP| + |LP|^2 + |LP| \cdot |TS|^2)$. If the simplifying assumptions are allowed, this reduces to $\mathcal{O}(S^2 + S \cdot |TS|^2)$.

ipcaEvaluation

Lines 2-4 evaluate the outcomes of the non-converged learner archives against their respective point archives ($|NC| \cdot |LA| \cdot |PA|$). Lines 5 to 12 evaluate the outcomes of the learner population against the point archives ($\mathcal{O}(|LP| \cdot |PA|)$) and evaluates usefulness on these outcomes ($\mathcal{O}(|LP| \cdot |LA| \cdot |PA|)$). Lines 13 to 23 introduce a double

for loop (over learners and points) which calls calcOutcomes and isUseful. The results of calcOutcomes are over the point archives and so can be cached so that only one new outcome is required on each iteration, making these calls approximately constant. The isUseful call requires $|LA| \cdot |PA|$ evaluations in this context, making the double loop complexity $\mathcal{O}(|LP| \cdot |PP| \cdot |LA| \cdot |PA|)$. Finally a call to updateArchives is required on line 24. Under the simplifying assumptions, the double for loop dominates the complexity with $\mathcal{O}(S^4)$

updateArchives

Lines 1 to 6 require $|NC|$ traversals of the useful points (size $|PP|$), each of which calls archivePoint ($\mathcal{O}(|PA| \cdot |P|)$). Additionally the learner outcomes are calculated against their respective archive ($\mathcal{O}(|LA| \cdot |PA|)$) for each element in NC for a combined complexity of $\mathcal{O}(|NC|(|PP| \cdot |PA| \cdot |P| + |LA| \cdot |PA|)$. If the simplifying assumptions are allowed, this reduces to $\mathcal{O}(S^2)$.

Lines 7 to 24 loop over the useful learners calling the following functions on each iteration:

1. calcOutcomes ($\mathcal{O}(|PA|)$)
2. isUseful ($\mathcal{O}(|LA| \cdot |PA|)$)
3. archiveLearner ($\mathcal{O}(|LA| \cdot |PA|)$)
4. calcOutcomes ($\mathcal{O}(|PP|)$)
5. min ($\mathcal{O}(|PP|)$)
6. archivePoint ($\mathcal{O}(|PA| \cdot |p|), p \in PA$))

This provides complexity of $\mathcal{O}(|LP|(|PA| + |LA| \cdot |PA| + |PP| + |PA| \cdot |p|)$. If the simplifying assumptions are allowed, this reduces to $\mathcal{O}(S^3)$.

stopCmge

The evaluation of early stopping criteria is performed over each non-converged class requires a call to rankHist which combines and re-ranks learner populations from

the current and previous epochs at a cost of $\mathcal{O}(|NC| \cdot |LP|^2 \cdot |Objective|)$. In the current framework, the number of objectives is small (4) and using the simplifying assumptions, the complexity reduces to $\mathcal{O}(S^2)$

buildSolution

The post training assembly and weighting of solutions is over the total number of solutions $c \cdot LA_{size}$, which additionally requires a loop over the training data, TD on lines 3-6 to find $\hat{n}$ and t for each class. Post training complexity is therefore in $\mathcal{O}(c \cdot LA_{size} \cdot |TD|)$

3.13.2 Time Complexity Analysis

Table 3.2: Algorithm Time Complexity Analysis

Algorithm	**Time Complexity**	**Discussion**
cmgeMain	See section 3.13.1, **cmgeMain**	See section 3.13.1, **cmgeMain**
initCmge	$\mathcal{O}(\|LP\| \cdot \text{CODONS})$	In practice the initialization of learners on line 2 dominates the complexity and this algorithm has the same complexity as initLearners()
trainCmge	$\mathcal{O}(S^2 + S \cdot \|TS\|^2)$	See section 3.13.1, **trainCmge**
ipcaEvaluation	$\mathcal{O}(S^4)$	See section 3.13.1, **ipcaEvaluation**
updateArchives	$\mathcal{O}(S^3)$	See section 3.13.1, **updateArchives**
stopCmge	$\mathcal{O}(S^2)$	See section 3.13.1, **stopCmge**
buildSolution	$\mathcal{O}(c \cdot LA_{size} \cdot \|TD\|)$	See section 3.13.1, **buildSolution**
initLearners	$\mathcal{O}(\|L\| \cdot \text{CODONS})$	Loops $\|L\|$ times over the number of codons, CODONS with each iteration being a constant call (random). Assumes the repeat ... until loop will map within a constant number of iterations.
fillPtPop	$\mathcal{O}(PP_{size})$	Assumes that TD can be sorted by class in advance (required once only), then loops over PP_{size}, each being constant assignments.
select	$\mathcal{O}(\|LP\|)$	Requires to loop twice over LP, once to find sum of population fitness and once to find where the random value falls.

Continued on next page

Table 3.2 Algorithm Time Complexity Analysis – continued from previous page

Algorithm	**Time Complexity**	**Discussion**
applyXover	$\mathcal{O}$ (CODONS)	Requires CODONS iterations to assign random codon values to $L1, L2$. Assumes the repeat ... until loop will map in constant iterations.
applyMutation	$\mathcal{O}$ ($\|L\| \cdot$ CODONS)	Requires CODONS iterations to assign random codon values $\|L\|$ times. In practice, $\|L\|$ is 2, so this is essentially $\mathcal{O}$ (CODONS). Assumes the repeat ... until loop will map in constant iterations.
geMap	$\mathcal{O}$ ($\|C\|$)	Requires the expansion of at most $\|C\|$ non-terminal symbols from the stack. Each expansion is a constant time lookup into G. In practice this may be repeated for a constant number of *wrap events* but does not affect complexity.
aDomB	$\mathcal{O}$ ($\|Objective\|$)	Requires two comparisons for each objective, therefore $O(\|A, B.Objective\|)$.
replace	$\mathcal{O}\left(\|LP\|^2 \cdot \|Objective\|\right)$	Requires a single pass of the learner population to find minimum rank in LP for each member of C (in practice this is two). Finally requires a call to rankLearners, which dominates the complexity of the linear time min function.
rankHist	$\mathcal{O}\left(\|LP\|^2 \cdot \|Objective\|\right)$	Requires a call to rankLearners, which is $\mathcal{O}\left(\|LP_c\|^2\right)$ where LP_c is at most $2 \cdot \|LP\|$ and thus rankLearners dominates the complexity.
potentialFn	$\mathcal{O}\left(\|X\|^2\right)$	Initially requires calculation of distance matrix (in practice these are all in one dimension, making the distance constant), therefore $\mathcal{O}\left(\|X\|^2\right)$. The second section of the algorithm can loop a maximum of $\|X\|$ times, removing one point per iteration and applying the potential decay process to all $\|X\|$ points.
dist	$\mathcal{O}$ ($\|P\|$)	Requires a single pass to calculate difference over all point dimensions, therefore $\mathcal{O}$ ($\|P\|$).
evalLearners	$\mathcal{O}\left(\|L\| \cdot \|TS\|^2 + \|L\|^2\right)$	Requires that for all learners in L, calls

Continued on next page

Table 3.2 Algorithm Time Complexity Analysis – continued from previous page

Algorithm	Time Complexity	Discussion
		assignLMF ($\mathcal{O}\left(|TS|^2\right)$); calcGpOut, wrapGpOut, and evalObjectives are also called but these are all linear with respect to $|TS|$. Finally calls rankLearners, which adds $\mathcal{O}\left(|L|^2\right)$.
evalArchives	$\mathcal{O}\left(|LA| \cdot |TS|\right)$	Two passes over LA for each element in TS.
calcGpOut	$\mathcal{O}\left(|TS|\right)$	One evaluation per point in TS
wrapGpOut	$\mathcal{O}\left(|TS|\right)$	One evaluation and two assignments for each point in TS as passed to calcGpOut.
assignLMF	$\mathcal{O}\left(|TS|^2\right)$	The call to potentialFn dominates the linear calculation of s over TS.
evalSse	$\mathcal{O}\left(|Y|\right)$	Requires a single pass over each label in L, output in Y.
rankLearners	$\mathcal{O}\left(|L|^2 \cdot |Objective|\right)$	Initializes ranks for a cost $\mathcal{O}\left(|L|\right)$ then tests dominance (aDomB) property between all pairs of learners. In principle this testing adds $\mathcal{O}\left(|L|^2 \cdot |Objective|\right)$ but can, in practice, be reduced to $\mathcal{O}\left(|L|^2\right)$.
test	$\mathcal{O}\left(1\right)$	Requires 1 random number.
random	$\mathcal{O}\left(1\right)$	Requires 1 calculation.
evalObjectives	$\mathcal{O}\left(|TS|\right)$	Requires call to evalSse, countTruePositive, and evalOverlap, which are all $\mathcal{O}\left(|TS|\right)$ in this context. Assume strlen is constant.
archivePoint	$\mathcal{O}\left(|P| \cdot PA_{size}\right)$	Assuming the point archives provide class-specific access, the selection of PA_c is constant. When the condition on line 6 fails, the a prune is required and the closest archive point must be found at a worst-case cost of $\mathcal{O}\left(|P| \cdot PA_{size}\right)$
archiveLearner	$\mathcal{O}\left(LA_{size} \cdot |outcome|\right)$	When the initial condition fails, a prune is required which calculates the minimum sum of outcomes over the archive size, $\mathcal{O}\left(LA_{size} \cdot |outcome|\right)$.
calcOutcomes	$\mathcal{O}\left(|L| \cdot |P|\right)$	Requires a call to calcGpOut and wrapGpOut ($\mathcal{O}\left(|P|\right)$, each) over each learner

Continued on next page

Table 3.2 Algorithm Time Complexity Analysis – continued from previous page

Algorithm	**Time Complexity**	**Discussion**
		in L. A constant assignment is additionally required for each element in P, however this is additive and therefore complexity remains $\mathcal{O}(\|L\| \cdot \|P\|)$.
isUseful	$\mathcal{O}(\|LS\| \cdot \|outcome\|)$	Requires evaluation of dominance property (aDomB) against the set of learners LS, each being $\mathcal{O}(\|Objective\|)$, where this evaluation considers learner outcomes as objectives.

Chapter 4

Benchmarking Methodology

Chapter three established the CMGE model for multi-class classification under a multi-objective competitive coevolutionary paradigm utilizing a local membership function. Needless to say, there are multiple differences relative to a canonical GE classifier. Our objective is now to design a benchmarking methodology that is able to incrementally demonstrate the utility of different elements of the CMGE framework, both with respect to related GP classifiers, and the wider machine learning literature. To this end we approach the problem from three perspectives:

1. Comparison against related GE models: We consider a total of three GE models incorporating incremental changes from canonical GE to a competitive coevolutionary model of GE, and two variants of CMGE implementing different pruning heuristics, as follows.

   (a) Standard GE (StdGE – Section 4.1.3): We begin by establishing a (stochastic learner) baseline in terms of a GE classifier trained over all exemplars in the training data, denoting such a model 'StdGE'. Our motivation here is to provide a reference point characterizing the classical canonical approach to GP classification. The canonical GE model is common to all GP classifiers studied in this thesis, and follows the algorithm detailed in Section 2.1 (or steps 5 to 18 of Algorithm 9).

   (b) Random (Balanced) Subset Selection (Rss GE – Section 4.1.4): The second GE model introduces class balanced random sampling of the training data to fill a subset of exemplars over which fitness evaluation takes place, RssGE. This represents the most straightforward approach for dealing with the computational overhead of GE, whilst avoiding the likelihood of degenerate solutions under unbalanced data sets (care of the equal class representation constraint).

(c) Pareto-coevolutinoary GE Classifier (PGEC – Section 4.1.5): Our third comparative algorithm adds the IPCA competitive coevolutionary model of de Jong (and shared by CMGE) to GE, resulting in the PGEC model. Benchmarking this model quantifies the difference between assuming CMGE's multi-objective model (with local membership function) as opposed to a GE classifier based on an SSE cost function calculated according to a sigmoid wrapper function. Note that both models return a set of solutions (the Pareto front). However, we maintain that the CMGE model is able to provide a much more accurate model for problem decomposition than PGEC on account of the local membership function and accompanying multiple objectives.

(d) Competitive Multi-objective GE (CMGE1 and 2 – Sections 4.1.1 and 4.1.2): Finally, we investigate the utility of two pruning heuristics under the CMGE paradigm, one purely greedy in this respect and the other taking class-specific error into account, resulting in CMGE1 and CMG2 variants of the generic CMGE model.

2. Comparison against other machine learning models requiring free parameter initialization: EC algorithms by their very nature require multiple initializations in order to establish sensitivity to the model initialization step[1]. We therefore compare GE classifiers to two widely utilized Neural Network (NN) models, denoting an example of a machine learning algorithm that is also sensitive to model initialization (in this case weight matrix), therefore requiring multiple model initializations per data partition. The NN paradigm is also interesting from the perspective that it generally assumes a rather different set of modeling biases than those assumed in EC. In particular, NN models generally assume a fully connected neurological basis for model building in which all features are explicitly indexed. Conversely, GP runs do not generally result in solutions that index all features, and have this property embedded as a basic search bias. Needless to say, there is potentially a wide range of NN models appropriate to

[1]We differentiate between parameters of the learning algorithm and free parameters of the model (i.e., representation).

classification problems with various different representations e.g., ART, RBF, MLP [48]. In this case we elect to concentrate on the popular paradigm of feedforward neural networks, a decision that lets us compare linear and non-linear variants of the same feedforward model.

(a) Linear Perceptron (LP – Section 4.1.6): The linear perceptron is a single layer architecture trained using a credit assignment policy based on gradient decent. As such, the model will 'descend' to the nearest local minima relative to the initial location in search space, where such locations are defined by the initialization of model parameters. Moreover, by assuming a single layer architecture, the model is limited to building solutions from linearly separable combinations of the input features. However, this in itself can be of significant utility when attempting to establish the advantage / disadvantage of non-linear models (such as GE). Finally, unlike previous studies in which NN methods have been benchmarked against GP classifiers we make use of an efficient second order policy for performing gradient decent. That is to say, comparative studies to date have tended to focus on feedforward models trained using first order gradient decent with momentum e.g.,, RPROP [15]. Such algorithms have widely been known to suffer from sensitivity to the initialization of model parameters, resulting in the utilization of a wide range of second order methods [69], [10].

(b) Multi-layer Perceptron (MLP – Section 4.1.7): This is a natural extension of the Linear Perceptron, the only model difference being the inclusion of a second layer of 'hidden' neurons between input features and layer of output neurons. All other aspects of the model remain unchanged relative to the linear perceptron. However, the addition of the hidden layer is sufficient to turn the MLP into a universal approximator [48], although the sensitivity to initial conditions has also increased on account of the additional model complexity and greedy credit assignment policy.

3. Comparison against deterministic machine learning models: A wide range of

machine learning models adopt a greedy model for model construction directly from the data set without assuming any biological motivation. Moreover, unlike NN and EC models they do not have a set of model parameters, thus are only sensitive to the composition of the data set (they will assume a set of learning parameters, such as pruning factors etc., but both NN and EC models also have such a set of learning parameters). For pragmatic reasons we limit ourselves to three examples, presented as follows with increasing complexity:

(a) OneR (1R – Section 4.1.8): The 1R algorithm takes the form of a single node decision tree, in effect classifying data based on the attribute most closely correlated with the class label. Despite the simplicity of this model, an early evaluation on data sets taken from the UCI repository indicated that the model would provide solutions within 5% of the much more sophisticated C4.5 algorithm [52].

(b) Naïve Bayes (NB – Section 4.1.9): The Naïve Bayes algorithm is a widely used statistical model based on prior probabilities estimated from the training data partition. Although comparatively 'simple' to formulate, the model is widely acknowledged to perform well on 'difficult' real world problems [31], [97].

(c) C4.5 decision tree (J48 – Section 4.1.10): C4.5 is the very widely utilized decision tree model of Ross Quinlan [96]. The model uses a greedy heuristic based on normalized entropy estimates to inductively construct a decision tree classifier. In common with OneR and Naïve Bayes it is very widely utilized in benchmarking studies [74] [1].

In order to compare the CMGE paradigm to a cross section of alternative machine learning algorithms, we are also interested in identifying models of classification that scale to large data sets, are equally at home in multi-class and binary domains, and are resilient to unbalanced distributions of class labels. In order to investigate the resilience of the CMGE paradigm to these factors it is necessary to explicitly identify data sets that demonstrate these properties to varying degrees. To this end we make use of previous benchmarking studies to explicitly seek out the more interesting data

sets [74], [117], [22], [15], as well as data sets explicitly compiled for international competitions [30]. Section 4.3 presents the twelve data sets identified, where eight are multi-class, and thee involve training data sets with over 10,000 exemplars.

Factors of interest include establishing the metrics used to qualify post training performance, and relevant statistical methodology. We therefore complete our benchmarking methodology by presenting our approach to evaluation in Section 4.4. The latter point is particularly important when attempting to compare EC or NN models against deterministic models. That is to say, deterministic models will produce a single solution per data partition. Even with 10-fold cross-validation this only results in ten performance points, where this is not sufficient to establish the basis for statistical tests. Conversely, EC and NN solutions require multiple runs per model initialization, resulting in fifty or more solutions per data partition. A framework is therefore necessary that enables us to compare single point solutions from a deterministic algorithm with sets of solutions from the EC paradigm. We address this problem by considering the likelihood of EC solutions matching or exceeding the performance point set by a deterministic method. Section 4.4 concludes by summarizing the approach to assessing solution 'coverage' or diversity, where this is an important attribute of learning models that present solutions in the form of an ensemble or team of classifiers (as is the case with CMGE and PGEC).

Finally, Section 4.6 summarizes the experimental design, where this consists of five sets of comparisons. In the first case various GE models are compared, beginning with canonical GE (iterates over the entire data set), and finishing with std GE versus RSS GE versus CMGE (all with a fixed evaluation limit) over all twelve data sets. Such an analysis is performed with respect to both accuracy and the more robust 'score' metric. A specific ranking of model performance results in which CMGE appears as a clearly superior model. CMGE is therefore retained for comparison against NN and the deterministic machine learning models, with RSS GE also reported to establish a GE performance base line. The remaining two experimental designs are used to investigate more specialist properties of CMGE. Specifically a 'coverage analysis' is established to demonstrate the different behavioral properties of the Pareto fronts. Lastly we review the sensitivity of CMGE to learning parameter selection.

4.1 Evaluation Limits and Methodology for Model Initialization

This section provides a summary of the algorithms employed in the benchmarking framework and describes the approach taken to establishing common computing limits. From the perspective of EC models this implies the utility of a common evaluation and initialization limits. In the case of ANN models initialization limits are required, whereas the deterministic (and therefore EC and ANN) models will require cross-validation in selective cases.

All (GE) systems were tested with an upper limit on the total number of individual evaluations. By holding constant the number of evaluations we are able to provide comparisons between algorithms such that learning efficiency per evaluation is taken into consideration. Specifically, the evaluation limit for each algorithm is defined as:

$$PP_{size} \times LP_{size} \times \text{MAX\_EPOCHS} \tag{4.1}$$

per class, where these values are supplied in Table 4.5. Each run of RssGE would therefore be permitted the number of evaluations defined in (4.1) and the algorithm run for each class of the data set, whereas the equivalent CMGE result would be from a single run that was permitted the maximum number of evaluations of (4.1) multiplied by the number of classes. Each stochastic experiment was run for 50 initializations. In the case of the two Neural Network models, we use the MATLAB® Neural Network toolbox [28] to model perceptron and MLP models, also under 50 initilizations (in this case referring to initialization of the 'weight' matrix). Deterministic systems are provided through the Waikato Environment for Knowledge Analysis (WEKA) machine learning software workbench[2], version 3.4.9 and include OneR (Section 4.1.8), Naïve Bayes (Section 4.1.9) and the C4.5 (release 8) decision tree (Section 4.1.10). The deterministic models assume a greedy algorithm for model building and are therefore only sensitive to the composition of the training data[3].

[2] http://www.cs.waikato.ac.nz/∼ml/index.html

[3] As such we will introduce the utility of ten fold cross-validation under the smaller data sets.

4.1.1 CMGE1

CMGE1 represents the Competitve Multi-objective GE described in the current work that employs a basic, greedy learner archive pruning strategy as follows: the learner archive member having the lowest sum of real-valued outcomes (against its class-appropriate archive) is replaced by the incoming learner. Here the outcome entries are defined for the i^{th} exemplar as follows:

$$\text{outcome}(i) = \begin{cases} y(gpOut) & \text{if class}(i) = \text{in-class} \\ 1 - y(gpOut) & \text{otherwise} \end{cases} \tag{4.2}$$

This pruning architecture is meant to encourage strong, accurate decisions among learner archive members. Moreover the pruning policy does not discriminate between the types of errors made by the classifiers (i.e., error that would contribute toward false positives vs. false negatives); when all else is the same, pruning therefore favors the learner archive members making strongly decisive classifications over sheer numbers of correct classifications as predicted by the binary ($\hat{o}$) outputs. This policy was designed to encourage survival of potentially cooperative team members by pruning individuals that respond weakly to archive points and may therefore cause conflicts in team-based classifications post-training.

4.1.2 CMGE2

CMGE2 represents the Competitve Multi-objective GE described in the current work that employs a two-stage greedy learner pruning policy as follows: the learner archive member having the lowest sum of real-valued outcomes (against its class-appropriate archive) across *out-of-class points* is replaced by the incoming learner if the new learner represents an improvement in this respect. Otherwise, the archive member having the lowest sum of outcomes is replaced as in CMGE1 (Section 4.1.1). This policy therefore discriminates against false positive error in the archive members by preferring to prune according to error type; however, when the incoming learner has a lower sum of out-of-class outcomes, the replacement policy reverts to the default policy of CMGE1, described in Section 4.1.1.

This pruning policy aims firstly to reduce the potential for false positive errors

while encouraging strong, class-consistent decisions in learner archive members regardless of the number of total correct classifications. This policy was designed to explicitly reduce the occurrence of false positives and thereby improve overall classification performance. Moreover, the policy also encourages the survival of potentially cooperative team members by pruning individuals that respond weakly to archive points and may therefore cause conflicts in team-based classifications post-training.

4.1.3 StdGE

StdGE is a standard canonical binary GE classifier implementation (Section 3.3) that employs a sigmoid wrapper function and performs fitness evaluation over all training exemplars (i.e., there is no subset selection mechanism; as such the learning follows Algorithm 1). As a binary classifier, each class requires a separate classifier expression, and a winner-take-all voting policy is therefore employed for consistency in comparisons. Such a policy represents a more strict model of classification than is typically used within the context of binary classification; however, such a model is implicit in the Neural Network classifiers and would be necessary for paradigms in which multiple labels are appropriate per class. Fitness under this model takes the form of a scalar SSE in conjunction with a sigmoid wrapper function evaluated over all training examples; adopting such an approach provides the basis for a more robust differentiation between classes than a switching function, as discussed in Section 2.2.1.

4.1.4 RssGE

RssGE provides a computationally tractable model of StdGE that employs a class-aware (balanced) version of the random subset selection (RSS) algorithm [42] in place of a point archive. RSS thus replaces the notion of point population and training set (PP and TS of Algorithm 6) and there is no concept of archives or early stopping in this algorithm (i.e., steps 4 and 5 of Algorithm 3 are removed). Fitness is based on a single metric that employs the error provided by the sigmoid wrapper evaluation, as per StdGE (Section 4.1.3). As a binary classifier, each class requires a separate classifier expression, and a winner-take-all voting policy is therefore employed for consistency in comparisons.

4.1.5 PGEC

The PGEC classifier is a basic GE implementation of Lemczyk's Pareto competitive coevolutionary Genetic Programming Classifier (PGPC) algorithm [70] [71], a multi-individual classification model with problem decomposition. Specifically the PGEC algorithm employs binary archiving where archive membership is driven by Pareto dominance on binary outcome vectors. Our deployment of PGPC utilizes a sigmoid wrapper function, replacing Lemczyk's switching wrapper (Figure 2.3, (b) vs. (a)). That is to say, a global sigmoid type wrapper is employed, as (unlike the CMGE model) there is no mechanism for guiding the development of suitable local membership functions and fitness is therefore evaluated on a single SSE metric (Algorithm 17) that employs the error provided by the sigmoid wrapper evaluation. We note, however, that the Pareto archiving criteria (binary outcomes) remain based on a switching wrapper at 0.5, as does the final majority voting policy. Early stopping is provided exclusively by the maximum number of evaluations specified in Equation 4.1. As a binary classifier, each class requires a separate collection of classifier expressions (meaning a separate run of PGEC per class), with the majority voting policy employed in combining the outputs.

4.1.6 Linear Perceptron (LP)

The Linear Perceptron model is configured as a single layer feed-forward backpropagation network of c (number of classes) nodes employing logarithmic sigmoid transfer functions and trained for 50 epochs using the Conjugate Gradient Method (CGM). That is to say we use an efficient second order method as opposed to the more common first order update rules, with the objective of reducing the sensitivity to local minima. As such, weights and bias terms are initialized to values over the interval [-1,1] with uniform probability. Weight and bias values are updated according to the conjugate gradient backpropagation algorithm with Fletcher-Reeves updates [69] [10]. Classification decisions are determined by the node having the maximum output value, $\max(y_o)$, $o \in \{1 \dots c\}$ where $y(o) = f(w \cdot x)$ and $f(\cdot)$ is the sigmoid activation function (tansig); $w \cdot x$ is the inner vector product between input exemplars and (neuron) weights. Input features (x_i) of the data set are subject to the following

normalization, where this is necessary to avoid biasing the weight update procedure in favor of features with larger dynamic ranges:

$$x_i^* = \frac{x_i - \mu_i}{\sigma_i} \tag{4.3}$$

where μ_i and σ_i are the mean and standard deviations of the i^{th} feature for the entire data set and x_i^* is the normalized feature value.

Finally as a sigmoid activation function is employed (Figure 2.3 (b)), class labels take the form of -0.9 and 0.9 for in and out-of-class exemplars, respectively. Such a procedure avoids setting the artificial objective of reaching target values of -1 and 1, which would require weight values that tend toward $\pm\infty$ on account of the asymptotic properties of the activation function.

4.1.7 Multi-layer Perceptron (MLP)

The Multi-layer Perceptron (MLP) extends the capabilities of the LP to perform non-linear mappings through the use of an intermediate layer of 'hidden' neurons. As such the resulting model has the proprieties of a universal function approximator [48]. The principal design decision now takes the form of determining the number of hidden layer neurons. Use of too many results in a 'memorization' behavior on the training data, whereas too few will result in failure to converge [48]. In order to arrive at our count of hidden neurons, we assume that the data sets described in terms of more features are likely to require more hidden layer neurons; however, we also employ a lower bound of six neurons, where this appears to represent a suitable minimum network complexity for the data sets considered[4]. The resulting Multi-layer Perceptron model is configured as a feed-forward backpropagation network with a hidden layer of size $\max(\lceil \frac{f}{2} \rceil, 6)$ and an output layer of c (number of classes) nodes. Each node employs the logarithmic sigmoid transfer function. As with the Linear model of Section 4.1.6, the network is trained for 50 epochs using the Conjugate Gradient Method (CGM) that updates weights and bias values according to the conjugate gradient

[4]Reference was also made to the Proben Neural Network benchmarking initiative when establishing our network configuration. Proben benchmarks the MLP architecture on various UCI data sets using the first order RPROP weight updating procedure [78].

backpropagation with Fletcher-Reeves updates [69] [10].

Conjugate gradient is a classical second order training model that provides robust performance relative to the traditional (first-order) gradient decent models; moreover, conjugate gradient methods are generally able reach convergence much faster than first order steepest decent directions used in the basic backpropagation algorithms [28] [69] [10]. The CGM update starts out by searching in the steepest direction (along the negative gradient) on its first iteration (Equation 4.4) using a line search to determine the optimal distance to move along the search direction (Equation 4.5) [28] [10].

$$\boldsymbol{p}_0 = -\boldsymbol{g}_0 \tag{4.4}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \cdot \boldsymbol{p}_k \tag{4.5}$$

Subsequent search directions are chosen to be conjugate to the previous direction by combining a weighting of the old direction with the new gradient (Equation 4.6). The calculation of the weight term β_k is by way of the Fletcher-Reeves update given in Equation 4.7, which provides the ratio of the norm squared of the current gradient to the norm squared of the previous gradient.

$$\boldsymbol{p}_k = -\boldsymbol{g}_k + \beta_k \cdot \boldsymbol{p}_{k-1} \tag{4.6}$$

$$\beta_k = \frac{\boldsymbol{g}_k^T \cdot \boldsymbol{g_k}}{\boldsymbol{g}_{k-1}^T \cdot \boldsymbol{g_{k-1}}} \tag{4.7}$$

Unlike most second order approaches which require the costly calculation of the Hessian matrix of partial derivatives ($\mathcal{O}(N^2)$) [10], the CGM remains in $\mathcal{O}(N)$ complexity while requiring only slightly more storage than the simpler algorithms [28]. The training efficiency and low storage complexity of this robust second order model motivate its use in the current work.

As per the LP model, the input features are normalized according to Equation

4.3, and outputs are normalized to avoid setting unreachable target values. The addition of the hidden layer of neurons does, unfortunately, make the model sensitive to initial conditions, on account of the non-linear properties of the network, introducing additional complexity into the search space. Fifty runs are therefore performed per data partition.

4.1.8 OneR

Robert Holte's OneR (1R or 1-Rule) algorithm is a single level decision tree which classifies data based on the attribute having the lowest error rate on the training data [52]. The algorithm is deterministic which assumes that class label and input features are correlated. In the early 1990s, Holte demonstrated that selecting the single most correlated feature was sufficient for providing performance competitive with the C4.5 algorithm on most data sets then available from the UCI repository [52][5].

4.1.9 Naïve Bayes (NB)

Naïve Bayes is a multi-class, statistical classifier that uses Bayes' theorem to model the conditional posterior (or *a posteriori*) probabilities for each of c classes: $P(C_i|X)$, where C_i is a hypothesis (e.g., class $= C_i$; $i = 1 \ldots c$) and X is an observation, or *evidence* in the form of an exemplar. The largest posterior probability for hypothesis C_i conditioned on X determines the label assigned by the classifier [45] [31].

Building the class-wise posterior probability models requires analyzing the training data to estimate *prior* probabilities, (or simply, *priors*), $P(C_i)$ along with probabilities of X conditioned on C_i ($P(X|C_i)$) and the basic probability of seeing exemplar X in the training data, $P(X)$. Bayes' theorem relates the conditional probabilities $P(C_i|X)$ and $P(X|C_i)$ by [45]:

$$P(C_i|X) = \frac{P(X|C_i) \cdot P(C_i)}{P(X)} \tag{4.8}$$

Given data with many attributes, the $P(X|C_i)$ term can be computationally expensive to estimate directly thus a simplifying (naïve) assumption is made that input features

[5]The C4.5 variant used in Holte's original comparison was an earlier version of the algorithm discussed in Section 4.1.10.

$(x_0 \ldots x_d)$ are conditionally independent given a class label. This reduces the estimate to [45]:

$$P(X|C_i) = \prod_{k=1}^{d} P(x_k|C_i) \tag{4.9}$$

The values $P(x_k|C_i)$ can now be estimated directly from training data[6] and with the posterior models in place, Naïve Bayes classifications take the form of:

$$\text{class} = \text{argmax}_i(P(C_i) \cdot \prod_{k=1}^{d} P(x_k|C_i)) \tag{4.10}$$

Despite widely acknowledged criticism of the 'naïve' independence assumption, Naïve Bayes models have been shown to perform very well in difficult real-world problems [31] [97]. The Naïve Bayes training process is deterministic in that it provides the same classifier for a given set of training exemplars. Comparisons with the GP models will be therefore in terms of the deterministic comparison method introduced in Section 4.5.3.

4.1.10 C4.5

C4.5 is a decision tree classifier based on the ID3 algorithm that is built by recursively splitting the training data based on greedy selection of the attribute providing maximum normalized information gain over the training set [96]. At each split, a conditional (if / then) node is introduced into the decision tree model based on the current attribute, creating branches that partition the data into subsets corresponding to each of the mutually exclusive outcomes. The algorithm is applied recursively on the non-uniform class partitions so that the i^{th} branch leads to the decision tree constructed from the corresponding data partition.

The information conveyed by a message is defined by the negative logarithm of the message probability. The expected information or *entropy* regarding class given a set of exemplars S is provided in Equation 4.11.

[6] If features are (e.g.,) categorical then $P(x_k|C_i)$ can be estimated directly as the number of exemplars of class C_i in TD having value x_k for attribute k divided by the total number of exemplars of class C_i in TD [45].

$$info(S) = -\sum_{j=1}^{k} f(S,j) \cdot log_2(f(S,j)) \text{ bits} \tag{4.11}$$

where

$$f(S,j) = \frac{|\{i \in S \mid \text{class}(i) = j\}|}{|S|} \tag{4.12}$$

A test X that partitions a training set TS by n outcomes into $TS_1 \ldots TS_n$ has entropy provided by Equation 4.13.

$$info_X(TS) = -\sum_{i=1}^{n} \frac{|TS_i|}{|TS|} \cdot info(TS_i) \tag{4.13}$$

Information gain (Equation 4.14) is calculated by taking the difference in entropy before and after the partitioning according to X.

$$gain(X) = info(TS) - info_X(TS) \tag{4.14}$$

In order to control for bias in favor of tests having many outcomes, the normalizing term of Equation 4.15 is used to divide the information gain by the potential information generated by dividing TS into n subsets [96]. Equation 4.16 provides the information gain ratio criterion for selecting tests in C4.5.

$$info_{split}(X) = -\sum_{i=1}^{n} \frac{|TS_i|}{|TS|} \cdot log_2\left(\frac{|TS_i|}{|TS|}\right) \tag{4.15}$$

$$gain_{ratio}(X) = \frac{gain(X)}{info_{split}(X)} \tag{4.16}$$

J48, the WEKA implementation of Quinlan's C4.5 (release 8) algorithm, is employed in the current work. The J48 classifier was run in batch mode using default parameter settings with no pruning.

4.2 Cross-validation

Cross-validation (or k-fold cross-validation) is a commonly used technique that involves splitting data into k (e.g., training / test) partitions to be used in repeated evaluation procedures such that subsequent analyses have less sensitivity to the bias inherent in partitioning. A k-fold cross-validation analysis begins with uniform random partitioning of the data into k stratified sub sets, or $folds$ (i.e., each fold $F_1 \ldots F_k$ contains $\frac{1}{k}$ of the data from each class and $F_1 \cap F_2 \cap \ldots \cap F_k = \{\varnothing\}$), as illustrated in Figure 4.1. On the first initialization, folds $F_2 \ldots F_k$ comprise the training data and F_1 (the *holdout*) is used for test; evaluations are then carried out using the typical procedures. The holdout for the i^{th} evaluation is then F_i, and the training set is comprised of the folds F_j $(\forall j \neq i)$. The distribution of the k evaluations is finally analyzed to provide a robust estimate of algorithm performance.

The current work employs ten fold cross-validation when no default partitions have been defined for the problem domain. This provides a training set comprised of 90% of the data with the remaining 10% being test on each of the 10 separate initializations. Moreover, the partitions are held constant across each of the algorithms employed in this work.

4.3 Data Sets and Partitioning

All experiments are undertaken using classification problems from the University of California at Irvine's (UCI) Machine Learning Repository [88]; two are specifically well-known as large and complex data sets from the UCI Knowledge Discovery in Databases (KDD) Archive [50]. All data sets were preprocessed by first taking the union of all partitions and mapping nominal attributes to integer values. Any repeating, incomplete or inconsistent records were then removed from the set and the train / test partitions (if pre-defined) were restored. Problems that did not pre-define training and test partitions were evaluated by ten fold cross validation, described above. Moreover, the same partitions of the data were used for all experiments, irrespective of the algorithm.

The 12 data sets for this work (Table 4.3) were chosen on the basis of number of

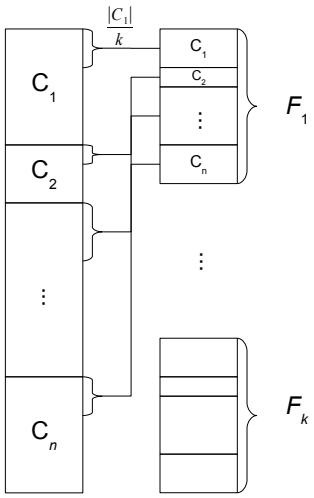


Figure 4.1: Partitioning of an n class problem for k-fold cross validation.

classes, number of features, number of exemplars, reported difficulty, and prominence in the literature so as to represent a diverse benchmark of binary and multi-class problems [74] [122] [55] [16] [116] [117] [22]. The data sets selected for this work, along with their general characteristics, relevant abbreviations and identifiers are summarized in Table 4.4. Below a brief discussion of each data set is provided in turn, including the range of previous performance results from the literature and any special considerations made for pre-processing / partitioning in the current work.

4.3.1 Boston Housing

The Boston Housing database is an approximately balanced, three class problem having 506 exemplars with 13 neumeric-valued attributes (not including the *medv* column). The data concerns housing values in the suburbs of Boston, MA. Classes are defined according to the *medv* column (median value of owner occupied homes in 1000s) as follows [74]:

- Class 1: $medv <= 9.84 \times 10^{-3}$;
- Class 2: $medv <= 10.075 \times 10^{-3}$;
- Class 3 otherwise.

There is no pre-defined partitioning of train and test sets for this problem, therefore ten-fold cross validation is applied.

The study of [74] reported overall training accuracy rates in the range of 75 to 78% with test accuracies in the literature (typically in terms of ten fold cross-validation) over a cross-section of approximately 40 machine learning algorithms providing accuracies in the range 63 - 78% [74] [40] [72].

4.3.2 Bupa Liver Disease

The Bupa Liver Disease database is an unbalanced binary (two class) classification problem containing 345 exemplars with 6 numeric attributes. There are four duplicate records and have been removed for the current work, leaving 341 exemplars. Of these, 142 (41.6%) belong to class 1 and the remaining 199 (58.4%) to class 2. There is no pre-defined partitioning of train and test sets for this problem, therefore ten-fold cross validation is applied.

This is generally considered a difficult problem domain with training accuracy in the region of 67-70% [40]; moreover, the data set has a history of use in the GP literature [55] [1]. Ten fold cross-validation results range from 54-72% (accuracy) [106] [9] [74], while very good results with arbitrary train / test partitioning have been reported in the range of 61-75% using GP approaches [87] [51] [72].

4.3.3 KDD: Census Income

The KDD Census Income database is an unbalanced binary (two class) classification problem containing 299,285 exemplars with 13 numeric and 28 nominal valued attributes. The data concerns the census statistics of total personal income, with the two classes being divided into those earning greater or less than \$50,000. The data has pre-defined partitions for train (199,523) and test (99,762). Among the training data, there are 3,229 duplicate exemplars that have been removed for the current work leaving 196,294 training exemplars. Of these, 183,912 (93.7%) belong to class 1 ($<$ \$50,000) and 12,382 (6.3%) belong to class 2 ($\geq$ \$50,000). Among the test data, there are 883 duplicate records that have been removed for the current work leaving 98,879 test exemplars. Of these, 92,693 (93.7%) belong to class 1 and 6,186 (6.3%)

belong to class 2.

The highly unbalanced (class-wise) nature of this data set make it a particularly challenging problem, with degenerate solutions being the norm; that is, test accuracies have been reported as high as 94-96% with detection rates (Equation 4.18) of 0.92-0.97, however this is typically offset by false positive rates (Equation 4.19) in the 0.79-0.85 range. [22].

4.3.4 Contraceptive Method Choice

The Contraceptive Method Choice database is an unbalanced three class classification problem containing 1,473 exemplars with 9 numeric attributes. There are 48 duplicate records that have been removed for the current work, leaving 1425 exemplars. Of these, 614 (43.1%) belong to class 1; 316 (22.2%) belong to class 2 and 495 (34.7%) belong to class 3. The data concerns prediction of the current contraceptive method choice (no use (class 1), long term (class 2) or short term (class 3)) of a woman based on her demographic and socio-economic characteristics. There is no pre-defined partitioning of data into train and test sets for this problem, therefore ten-fold cross validation is applied.

The study of [74] reported accuracies in the range of 40-57% over the 33 algorithms tested, which is largely consistent with the GP literature where test accuracies are frequently reported between 43 and 57% [40] [9].

4.3.5 Image Segmentation

The Image Segmentation database is a balanced, seven class classification problem containing 2310 exemplars with 19 numeric attributes. The data concerns the classification of 3×3 pixel segments taken from seven digital images of outdoor scenes: brick, sky, foliage, cement, window, path and grass, for classes 1 through 7 respectively. The data has pre-defined partitions for train (210) and test (2100), which are employed again here. There are 30 exemplars in each class of training, while in the test data there are 14 duplicate exemplars (3 of class 1; 1 of class 3; 2 of class 5 and 8 of class 6) that have been removed for the current work leaving 2086 test exemplars.

The 33 algorithms tested in the study of [74] reported accuracies in the range of

48-98%. Notably, however, only two algorithms were able to achieve results within one unit of standard error of the best result.

4.3.6 Iris Plant

Fisher's Iris Plant database is a balanced three class classification problem containing 150 exemplars with four numeric attributes: sepal length, sepal width, petal length and petal width. The data concerns the classification of Iris plants into one of three classes: Iris Setosa (class 1), Iris Versicolour (class 2), and Iris Virginica (class 3). One class is linearly separable from the other 2 but the latter are not linearly separable. Among the data, 3 duplicates exemplars have been removed for the current work, leaving 147 exemplars. There is no pre-defined partitioning of train and test sets for this problem, therefore ten-fold cross validation is applied.

Fisher's Iris data set is one of the most widely referenced in the machine learning literature and is generally considered to be a straightforward problem as compared to (e.g.,) Liver. The GP literature routinely reports test accuracy results in the range of 95-100% [72] [9] [93] [75].

4.3.7 KDD: KDD 99 Cup

The KDD 99 (KDD Cup) database was taken from the Third International Knowledge Discovery and Data Mining Tools Competition from KDD-99, The Fifth International Conference on Knowledge Discovery and Data Mining [50]. The KDD Cup competition involved generating a classifier able to discriminate 'normal' from 'attack' (intrusion) network connection types. The data set has been configured as a five class problem by categorizing each of the original 40 nominal connection type labels into 'normal' or one of four attack types[7]. Appendix H provides a full listing of the original nominal descriptors with associated class labels. The original training data set has 41 attributes (3 nominal, 38 numeric), with pre-defined partitions for train (494,020) and test (311,029). Among the training data there are 348,436 duplicates, while test contains 233,742 duplicate exemplars which have been removed for

[7]Details of attack definitions can be found at MIT Lincon Laboratory web site: http://www.ll.mit.edu/IST/ideval/data/data_index.html

the current work, leaving 145,584 train and 77,287 exemplars for test. The data set is highly unbalanced with the majority of exemplars associated with class 1; moreover, the class distribution differs between training and test sets. The characteristics of the train and test distributions are summarized in Table 4.1.

Results on KDD 99 have recently been reported for an unsupervised learning context between 67-71% accuracy, with 'scores' (Equation 4.21) of 0.55 and 0.60, respectively [57]. The winning result of Pfahringer corresponded to 92.7% accuracy (a 'score' of 60.4) [30]. Moreover, the 1-nearest neighbor approach is known to perform very well on KDD with test results approaching those of Pfahringer (92.3% accuracy, and 'score' of 55.2) [30].

Table 4.1: Processed KD99 data set characteristics (class-wise train / test exemplar distributions)

Class	**Train**	**%**	**Test**	**%**
1	87,831	60.33	47,913	61.99
2	999	0.69	3,058	3.96
3	54,572	37.48	23,568	30.49
4	2,130	1.46	2,678	3.47
5	52	0.04	70	0.09

4.3.8 Pima Indians Diabetes

The Pima database is an unbalanced binary classification problem containing 768 exemplars with 8 numeric attributes related to the task of diabetes diagnosis according to World Health Organization criteria. Of these, 500 exemplars (65%) are normal (class 0), and 268 (35%) are positive for diabetes (class 1). There is no pre-defined partitioning of the data into train and test sets for the problem, therefore ten-fold cross validation is applied.

The study of [74] reported test accuracies in the range of 69-78% for the Pima data set using ten fold cross-validation. Test results in the GP literature (typically involving an arbitrary partitioning into train / test sets) have been reported in the range of 68-80% [15] [106].

4.3.9 Statlog Project: Shuttle

The UCI Statlog Shuttle data set is a highly unbalanced seven class problem with 9 numeric attributes and a total of 58,000 exemplars. The problem has pre-defined partitions for train (43,500) and test (14,500) with approximately 80% of all data coming from class 1, while as few as 6 patterns are provided for class 6 in the training data. The characteristics of the train and test distributions are summarized in Table 4.2.

Table 4.2: Processed SHUT data set characteristics (class-wise train / test exemplar distributions)

Class	**Train**	**%**	**Test**	**%**
1	34,108	78.41	11,478	79.16
2	37	0.09	13	0.09
3	132	0.30	39	0.27
4	6,748	15.51	2,155	14.86
5	2,458	5.65	809	5.58
6	6	0.01	4	0.03
7	11	0.03	2	0.01

4.3.10 Thyroid Disease

The Thyroid Disease ANN database is a highly unbalanced three class classification problem containing 7200 exemplars with 21 numeric attributes. The data concerns the medical diagnosis of Thyroid Disease; the problem is to determine whether a patient referred to the clinic is hypothyroid. The three classes are defined as follows: class 1 is normal (not hypothyroid), class 2 is hyperfunction and class 3 is subnormal functioning [88]. Among the training data, there are 63 duplicate exemplars that have been removed for the current work leaving 3709 training exemplars. Of these, 93 (2.5%) belong to class 1, 191 belong to class 2 (5.1%) and 3425 (92.3%) belong to class 3. Among the test data, there are 8 duplicate records that have been removed for the current work leaving 3420 test exemplars. Of these, 73 (2.1%) belong to class 1, while 177 belong to class 2 (5.2%) and 3170 (92.7%) belong to class 3. The data

set is naturally of interest to this work on account of the widely unbalanced class distributions.

This data set has a history of use in the machine learning literature [74] and was employed in a simplified binary form by Gathercole and Ross in the original formulations of the RSS / HSS / DSS subset selection algorithm [42] [15]. The 33 algorithms tested in the study of [74] reported accuracies in the range of 11-98%, with typical figures from GP literature indicating results in the 95-98% range [15] [75].

4.3.11 Wine Recognition

The Wine Recognition database is an unbalanced three class problem containing 178 exemplars with 13 numeric numeric attributes related to the chemical analysis of wines grown in the same region in Italy. The task involves classification of the exemplars according to the cultivar from which they were derived. Class 1 is represented by 59 exemplars (33%), class 2 has 71(40%) and class 3 has 48 (27%). There is no pre-defined partitioning of the data into train and test sets for the problem, therefore ten-fold cross validation is applied.

The Wine Recognition data set is frequently employed in the GP classification literature where ten fold cross-validation results in the range of 92-98% are not uncommon [9] [106].

4.3.12 Wisconsin Breast Cancer

The Wisconsin Breast Cancer database (also known as Breast Cancer, Original) is a binary classification problem that contains 699 exemplars with 9 numeric-valued features per pattern. This is an unbalanced data set, with 244 patterns of class 1 and 455 of class 0. The data has 16 exemplars with missing attributes and 8 repeated exemplars that were removed for the current work, leaving 675 exemplars in total. Of these, 439 (65%) belong to class 1 and 236 (45%) belong to class 2. There is no pre-defined partitioning of the data into train and test sets for the problem, therefore ten-fold cross validation is applied.

The Wisconsin Breast Cancer data set is widely used in the literature [55] [15] [74] but in comparison to the (e.g.,) Liver data set (Section 4.3.2), it is much easier to

classify. The ten fold cross-validation results presented in [74], for example, indicate frequent results in the range 96-97%; however both training and test accuracies have been reported at 98-99% in recent GP literature [87] [40].

Table 4.3: Data set abbreviations

ID	DB Name	Abbreviation
D1	Boston Housing	BOST
D2	BUPA Liver Disease	BUPA
D3	KDD: Census Income	CENS
D4	Contraceptive Method Choice	CONT
D5	Image Segmentation	IMAG
D6	Iris Plant	IRIS
D7	KDD: KDD 99 Cup	KD99
D8	Pima Indians Diabetes	PIMA
D9	Statlog Project: Shuttle	SHUT
D10	Thyroid Disease (ANN)	THYD
D11	Wine Recognition	WINE
D12	Wisconsin Breast Cancer	WISC

4.4 Performance Measures

4.4.1 Classification Performance

Given the unbalanced nature of the data sets we make extensive use of a combination of post-training performance evaluation metrics. Moreover, we chose metrics that measure 'orthogonal' properties, in the case of Detection Rate and False Positive Rate, to provide additional insight into the resulting models. Accuracy is included as it is still one of the most widely used, albeit not particularly informative, metrics in the wider literature. We also adopt class-wise reporting of performance, following an initial evaluation in terms of accuracy, assuming median and quartile reporting in performance to the more biased mean and standard deviation based statistics. The three performance metrics are defined as follows, in terms of true positive (tp), true negative (tn), false positive (fp) and false negative (fn) [54]:

Table 4.4: Data set characteristics: exemplar counts (Total, Train, Test), number of features (F), number of classes (C) and class distributions (% Class Distribution).

DB	Total	Train	Test	F	C	% Class Distribution
BOST	506	10-fold	10-fold	13	3	$\sim$ Balanced
BUPA	341	10-fold	10-fold	6	2	42:58
CENS	295,173	196,294	98,879	41	2	94:6
CONT	1,425	10-fold	10-fold	9	3	43:22:35
IMAG	2,310	210	2,086	19	7	$\sim$ Balanced
IRIS	147	10-fold	10-fold	4	3	$\sim$ Balanced
KD99	222,871	145,584	77,287	41	5	Table 4.1
PIMA	768	10-fold	10-fold	8	2	65:35
SHUT	58,000	43,500	14,500	9	7	Table 4.2
THYD	7,129	3,709	3,420	21	3	2:5:93
WINE	178	10-fold	10-fold	13	3	33:40:27
WISC	675	10-fold	10-fold	10	2	65:45

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{4.17}$$

$$Sensitivity = DR = \frac{tp}{tp + fn} \tag{4.18}$$

$$FPR = \frac{fp}{fp + tn} \tag{4.19}$$

Finally, we also adopt a paired 'score' metric (Equation 4.21) which evaluates the equally weighted combined behavior of a candidate solution. In doing so we are able to capture whether the classifiers with good detection (sensitivity) are also capable of providing high specificity (Equation 4.20). In an n-class scenario, score is (more generally) calculated as an equal weighting of the class-wise detection rates.

$$Specificity = \frac{tn}{tn + fp} \tag{4.20}$$

$$Score = \frac{Sensitivity + Specificity}{2} = \frac{DR_1 + DR_2 + \ldots + DR_n}{n} \tag{4.21}$$

The above 'score' based policy is adopted in preference to the more general ROC / AUC approach for pragmatic reasons. The ROC / AUC metric builds a complete picture of pairwise 'orthogonal' performance metrics, as discussed above, by effectively thresholding a classifier at different intervals between two performance extremes (all examples in vs. all examples out-of-class). This is a relatively straightforward procedure for binary single model solutions (as in the 'super individual' returned under a canonical GP classifier). However, we have solutions comprised of tens of classifiers (in the case of CMGE and PGEC) with local and global activation functions. As such, there are now multiple possibilities for thresholding the overall model to produce the ensuing ROC curve. Examples including establishing ROC curves per individual participating in a solution to taking a group of individuals associated with each class and subjecting them to a group thresholding process. Clearly, the process of building such curves introduces a series of tradeoffs in the reported AUC. In addition to this the computational cost of building such an analysis over data sets consisting of tens or hundreds of thousands of exemplars also becomes prohibitive, especially given the number of model initializations / model types reviewed. In short, the combination of the three metrics (plus overall accuracy) represents our compromise solution adoped on the ensuing benchmark study for reporting / analysis of post-training classifier performance.

4.4.2 Computational Overhead

Total (processor) run time will be reported in terms of number of seconds elapsed as reported by the UNIX *time* utility. Specifically the value returned for the user time (the processor time charged for the execution of user instructions) will be considered as the total run time. All times include time to initialize, train and test the GP models[8].

[8]See the Unix (Darwin) manual page for *time*.

Table 4.5: Parameter Specifications

Grammatical evolution	
MAX_EXP_LEN	4096
CODONS	256
MAX_EPOCHS	500
Clustering	
α	150
β	155
γ_{lower}	0.5
γ_{upper}	0.75
Archives and populations	
LP_{size}	50
LA_{size}	30
PP_{size}	30
PA_{size}	30
Crossover and mutation	
PXO, PCXO	0.50, 0.90
PM, PTSM	0.01, 0.90
Early stopping	
MIN_DIFF	0.1
CONV_MIN_POP	20
CONV_FRAC	$\frac{1}{5}$

4.4.3 Solution Complexity

Solution sizes are reported in terms of median string length of the unsimplified expression. In the case of models providing solutions in terms of multiple individuals (CMGE and PGEC), we report median and quartile lengths over the individual models contributing to the solution.

4.4.4 Significance Measures

Notched box plots are used to summarize results in terms of median, first and third quartiles as well as maximum and minimum values. The horizontal line in the middle of a box indicates the median while the lower and upper lines of the box represent the first and third quartiles, respectively with the inter-quartile range being the difference between these values. Box notches about the median represent a confidence interval; non-overlapping notches between a pair of results indicate a difference of statistical significance at a 95% confidence level. Whiskers (dashed lines extending beyond the top and bottom of the box) indicate the minimum and maximum values in the data, unless outliers are present. Outliers are defined as points being 1.5 times the interquartile range away from the quartiles.

To confirm the statistical significance of differences between the groups of results the Analysis of Variance (ANOVA) F test procedure is first applied to test the null hypothesis that all groups come from distributions having the same means (i.e., $H_0 : \mu_1 = \mu_2 = ... = \mu_c$) [8]. ANOVA models assume independence of samples as well as normality and equal variance of the underlying distributions [8]. The latter two of these assumptions may be in question; consequently, ANOVA results will be provided based on F test as well as the non-parametric (Kruskal-Wallis H test) version of ANOVA, which tests the null hypothesis that all groups have the same probability distribution against the alternate hypothesis that at least two of the distributions differ in location [83]. The advantage of the Kruskal-Wallis H test over the F test is that it is based on the analysis of variance using the ranks of values rather than the values of the samples themselves and therefore no assumptions need to be made about the shape of the sampled populations [83].

Once the initial ANOVA tests are performed, an *a posteriori* multiple comparison

procedure is applied using Tukey's honestly significant difference criterion based on the Studentized range distribution [8]. Multiple comparison plots provide graphical representation of pairwise comparison results between means and mean ranks at the 95% confidence level. Within the multiple comparison plots, pairwise groups having vertically non-overlapping bars have a statistically significant difference in terms of group means and rank means. For the purposes of the current work, significance will be indicated by agreement in both ANOVA and Kruskal-Wallis based multiple comparison results.

4.4.5 Coverage Measures

Coverage plots are class-grouped histograms that indicate the distribution of difference in vote strength between the winning output and the median output of the other individuals having the same class. In this way a difference value is calculated for each exemplar over the relevant number of partitions and initializations. The histogram uses twenty bins to display the difference values (between zero and one). A difference of zero indicates no difference between winner and median case of other in-class voters – i.e., a large degree of overlap, while a difference of 1 indicates the maximum difference between winner and median case of other in-class voters – i.e., a low degree of overlap.

4.5 Experiments

For the current work an initialization is defined as a single run of a classifier algorithm against a particular data set, while an experiment represents a set of one or more initializations of a classifier algorithm over a collection of data sets. All experiments were run on commodity-class Apple (iMac, Mac Pro or Xserve) dual-processor hardware with between 1 and 2 Gigabytes of RAM and running the Mac OSX Tiger operating system.

Twelve experiments were designed to establish comparative results over the set of problems in terms of classification accuracy, scalability, problem decomposition, and solution complexity. Generally speaking, the results of five experiment types (outlined in Table 4.6) are presented in this research.

4.5.1 Direct comparisons

Experiments E1 to E4 (Table 4.6) provide a direct means to compare the results of the current classifier with other classification approaches considered standard in the GP context. Classification performance will be established in terms of Accuracy, False Positive Rate, Detection Rate (Sensitivity), Specificity and Score. The significance of the comparisons between algorithms will be established using notched box plots along with multiple comparison analysis based on one-way ANOVA and its non-parametric counterpart, the Kruskall / Wallis analysis (see Section 4.4.4).

4.5.2 Artificial Neural Network (ANN) comparisons

Experiments E5 to E6 in Table 4.6 provide a means to compare the results of the current (CMGE) classifier with the standard Linear Perceptron (LP) in Experiment E5 and non-linear, multi-layer perceptron (MLP) in Experiment E6. Two Neural Network (NN) models trained using the conjugate gradient algorithm are used to establish the performance baseline, the motivation being:

1. Efficient credit assignment: Credit assignment in NN is typically based on some form of gradient optimization algorithm. The original first order methods popularized by the back propagation algorithm were very sensitive to local minima (weight initialization). Conversely second order methods have proved to be much more robust [69] [10]. However, a trade off exists between accuracy and computational expense, with the most accurate second order gradient optimization routine of the LM algorithm having a complexity of $\mathcal{O}(N^3)$ (with N being the number of training exemplars). This work therefore employs the conjugate gradient approximation to provide a linear complexity whilst maintaining a second order model [69] [10].

2. Linear versus non-linear: Both single layer (perceptron or linear) and hidden layer (non-linear) NN are built. The training algorithm remains unchanged (conjugate gradient), thus providing baselines for both linear and non-linear models trained over the same paradigm.

Classification performance will be established on the basis of Accuracy, False Positive Rate, Detection Rate (Sensitivity), Specificity and Score. The significance of the comparisons between algorithms will be established using notched box plots along with multiple comparison analysis based on one-way ANOVA and its non-parametric counterpart, the Kruskall / Wallis analysis (Section 4.4.4).

4.5.3 Deterministic comparisons

GPs and NNs are sensitive to the model parameter initialization. As such both NN and GP models require evaluation over a set of initializations; weight values in the case of NNs and populations in the case of GP. The same is not necessarily true of all machine learning algorithms, such as decision trees. Experiments E7 to E9 in Table 4.6 therefore provide a means to compare the results of the current classifier with these standard Deterministic approaches to classification, introduced in Section 4.1, specifically OneR in Experiment E7, Naïve Bayes in Experiment E8 and C4.5 in Experiment E9.

The procedure that we adopt for the deterministic comparison involves first establishing the baseline result statistic r_b in terms of the deterministic algorithm in question. The stochastic algorithm is then run over a set of N initializations, collecting the relevant statistic for each run in $r_s[i]$ $(i \in \{0 \ldots N-1\})$. This set of stochastic results is then analyzed in terms of the set of points falling at or above (R_{gte}) and below (R_{lt}) the single baseline result (r_b) returned by the deterministic algorithm[9]:

$$R_{gte} = \{i \in \{0 \ldots N-1\} \mid r_s[i] \geq r_b\} \tag{4.22}$$

$$R_{lt} = \{i \in \{0 \ldots N-1\} \mid r_s[i] < r_b\} \tag{4.23}$$

The probability of an equal or improved result, relative to the baseline is then the fraction of points falling at or above the baseline ($|R_{gte}|$) to the total number of initializations (N):

[9]In the case of (e.g.,) false positive rate, a lower result indicates improvement and these expressions must be modified accordingly.

$$P(r_s \geq r_b) = \frac{|R_{gte}|}{N} \tag{4.24}$$

Similarly, the probability of a result that performs worse than the deterministic baseline is given by:

$$P(r_s < r_b) = \frac{|R_{lt}|}{N} \tag{4.25}$$

Such a procedure is adopted in favor of the comparison of distributions used in the case of EC and NN comparisons on account of the implicit disjunction in the number of solutions. That is to say, thirty points is generally assumed to be the minimum number of points necessary to establish a distribution [19]. However, conducting thirty fold cross-validation on fifty EC (or NN) initializations per fold is clearly not a feasible experimental design. Moreover, this does nothing to address the disparity in the number of solutions that require comparison. Conversely, the above approach addresses what we feel to be the real question when comparing stochastic methods such as EC to strictly deterministic or greedy machine learning algorithms. That is to say, EC machine learning is built on the premise that the stochastic model has the 'potential' to identify solutions that a deterministic model would never encounter. The above metric is specifically designed to answer how likely the stochastic model is to perform equally or better as a result of the ability to make alternative decisions during model building.

The results of Experiments E7 to E9 (Table 4.6) will permit comparisons between the current classifier and deterministic algorithms on the basis of Accuracy, False Positive Rate, Detection Rate (Sensitivity), Specificity and Score statistics (Section 4.4).

4.5.4 Coverage analysis

The characterization of the pattern coverage achieved through problem decomposition will be established based on Experiments E10 and E11(Table 4.6). Specifically, both PGEC and CMGE return a 'front' of solutions per class. As such we are interested in determining how distinct these solutions are relative to other members of the same

front. To do so, a 'coverage' metric is employed that addresses the difference in the winning output and the median of other in-class outputs on each exemplar. Thus if the winning classifier has a behavior shared by the majority of the other individuals in the Pareto front, then a 'weak learner' or ensemble type of interaction will have taken place. In contrast, if the degree of classifier specialization is high / unique, then a larger difference will be measured relative to the median behavior of the Pareto front of classifiers. Coverage is plotted as a histogram of difference values, as described in Section 4.4.5.

4.5.5 Parameter analysis

Experiment E12 (Table 4.6) represents a series of initializations over three data sets intended to investigate the key parameters driving the CMGE algorithm complexity and classification performance, namely the learner and point archive sizes. Five settings were chosen for each archive parameter (10 to 50 inclusive, in increments of 10) and all 25 possible combinations were tested for 30 initializations on each of the three data sets. Surface plots indicate the results in terms of Accuracy, False Positive Rate, Detection Rate (Sensitivity), Specificity and Score (Section 4.4).

4.6 Summary

A multi-faceted experimental design has been proposed in which we incrementally establish the contribution of different components of EC models to the proposed CMGE classifier. Moreover, we consider methodologies for comparison against ANN and deterministic machine learning models. Relative to previous EC benchmarking research, we utilize a more sophisticated ANN learning algorithm, and recognize the difficulty in directly comparing EC to deterministic models directly. Finally, we make use of previous benchmarking studies to identify a wide cross-section of data sets that are illustrative of the many facets of real-world data sets, such as unbalanced exemplar distributions, non-linear interactions between features, and large exemplar counts.

Table 4.6: Experiment abbreviations

Direct (GE) Comparisons			
ID	**Algorithm**	**Inits**	**Data Sets**
E1	CanGE	50, 20	D10, D3
E2	RssGE	50	D1-D12
E3	StdGE	50	D1-D12
E4	CMGE1	50	D1-D12
E5	CMGE2	50	D1-D12
ANN Comparisons			
ID	**Algorithm**	**Inits**	**Data Sets**
E6	LP	50	D1-D12
E7	MLP	50	D1-D12
Deterministic Comparisons			
ID	**Algorithm**	**Inits**	**Data Sets**
E8	OneR	1	D1-D12
E9	NB	1	D1-D12
E10	C4.5	1	D1-D12
Coverage Analysis			
ID	**Algorithm**	**Inits**	**Data Sets**
E11	PGEC	30	D1-D12
E12	CMGE1	30	D1-D12
Parameter Analysis			
ID	**Algorithm**	**Inits**	**Data Sets**
E13	CMGE1	30×25	D1, D4, D10

Chapter 5

Results of Direct Comparisons

Experiments E1-E5 provide results that establish the baseline performance characteristics the GP classifiers employed in the current work. The following sections will first analyze the canonical form of GP (CanGE) with respect to performance against the four 'scalable' models. The results of experiment E1 will demonstrate the pathologies of the classical approach to multi-class classification using multiple class-wise runs of the traditional form of GP. Specifically, the canonical GP classifier will be shown to require multiple runs, each involving prohibitive training time while returning solutions that are significantly more complex and typically degenerate in the face of the class imbalance problem. Moreover, the monolithic solution form (employing a global wrapper function) will be discussed and provide motivation for the local membership function and problem decomposition approach employed by the proposed CMGE framework. The results of experiments E2 - E5 will provide a preference ordering of the scalable models in terms of classification performance, scalability, and solution complexity. A significant preference for the Competitive Multi-objective GE framework will be established across each of the comparison metrics.

A brief discussion of multi-class PGEC performance will also be provided; however, we do not examine these results in detail as the PGEC algorithm was originally tailored to the binary classification context and our multi-class variant was not generally competitive with the other multi-class GP approaches considered in this work. The complete set of PGEC classification results are, nonetheless, supplied in Appendix G.

Fitness assignment on all comparative GP models is based on an SSE evaluation in conjunction with a sigmoid wrapper function, while the CMGE algorithms employ a multiobjective evaluation scheme and a local (Gaussian) membership function. In the case of CanGE and StdGE, the fitness evaluation is over the entire training data set, whereas the RssGE, PGEC and CMGE models perform fitness evaluation

Table 5.1: CAN-GE - CENS Quartile Summary

Overall Score (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.500 / 0.500	0.500 / 0.500	0.531 / 0.531
Overall Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.937 / 0.937	0.937 / 0.937	0.940 / 0.941
Training time (s)			
C	**Q1**	**MED**	**Q3**
-	139020.1	172408.7	269413.7
Classwise Soln Size			
C	**Q1**	**MED**	**Q3**
1	12.0	27.5	78.5
2	13.0	41.5	67.5
Classwise Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.937 / 0.937	0.937 / 0.937	0.940 / 0.941
2	0.937 / 0.937	0.937 / 0.937	0.940 / 0.941
Classwise Det Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.998 / 0.998	1.000 / 1.000	1.000 / 1.000
2	0.000 / 0.000	0.000 / 0.000	0.063 / 0.063
Classwise FP Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.937 / 0.937	1.000 / 1.000	1.000 / 1.000
2	0.000 / 0.000	0.000 / 0.000	0.002 / 0.002

over a training subset. Notably, none of these approaches employs the post training performance metrics (e.g., accuracy, score) in the training algorithm.

5.1 Canonical GP Results

The large and unbalanced Census and Thyroid data sets provide the basis for the canonical multi-class GP experiments undertaken in the current study. These problems were selected for their size and traditional difficulty as unbalanced problems in order to clearly demonstrate the classical pathologies associated with the traditional approach to multi-class classification in the GP context.

5.1.1 Classification Performance

Experiment E1 Census-Income quartile summary results are provided in Table 5.1. In terms of overall classification performance, an obvious contrast exists between the Overall Accuracy and Score results with a substantially larger result in accuracy, while the score result indicates a guessing or degenerate classification behavior with very little variation in this result. The accuracy / score disparity is present over all three quartiles on both training and test results, which is highly indicative of degenerate solutions. This is readily confirmed using Table 5.1 by examining the class-wise Detection Rate and False Positive rate rows, where both train and test results achieve near unity on all three of the class 1 detection rate quartiles, while attaining approximately zero detection on class 2 (aside from quartile 3 which reaches a nominal performance of approximately 6.3%). Similarly, the class-wise false positive results reach unity on on all three quartiles of class 1 (indicating the worst possible FPR performance), while there are no false positives recorded for class 2 over the first two quartiles.

Being an unbalanced, binary data set with approximately 93% of data belonging to class 1, and 6% of data belonging to class 2, the Census-Income problem clearly demonstrates the canonical GP's reluctance to train beyond degeneracy in the face of a substantial class imbalance. This phenomenon is straightforward to explain; being that the canonical model is trained with predominantly class 1 data, the error function reaches a local minimum when a simple global 'guessing' behavior (in favor of class 1) is evolved. In the event that new individuals appear with a trade-off behavior (correctly identifying some small fraction of class 2 in place of the entirety of class 1), the error associated with missed class 1 exemplars will now outweigh the error gained by correctly identifying the exemplars of class 2, resulting in a relatively low fitness assignment and therefore a low probability of selection. In other words training gradient for this behavior is not readily established. Needless to say, the accuracy metric completely fails to reflect this behavior [54], thus the discrepancy is specifically recorded in the discrepancy between 'score' and accuracy metrics.

E1 Thyroid quartile summary results are provided in Table 5.2. The classification results demonstrate a trend similar to that of E1 Census-Income. Though the result is

Table 5.2: CAN-GE - THYD Quartile Summary

Overall Score (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.441 / 0.438	0.625 / 0.610	0.718 / 0.706
Overall Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.933 / 0.933	0.944 / 0.942	0.957 / 0.953
Training time (s)			
C	**Q1**	**MED**	**Q3**
-	5241.0	7148.2	8529.2
Classwise Soln Size			
C	**Q1**	**MED**	**Q3**
1	25.0	54.5	112.0
2	29.0	72.0	111.0
3	36.0	67.0	116.0
Classwise Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.980 / 0.983	0.989 / 0.989	0.991 / 0.990
2	0.949 / 0.948	0.950 / 0.949	0.960 / 0.958
3	0.939 / 0.940	0.951 / 0.947	0.969 / 0.960
Classwise Det Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.183 / 0.247	0.715 / 0.678	0.806 / 0.795
2	0.000 / 0.000	0.102 / 0.082	0.524 / 0.469
3	0.996 / 0.992	0.999 / 0.996	0.999 / 0.997
Classwise FP Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.000 / 0.001	0.002 / 0.003	0.004 / 0.006
2	0.000 / 0.000	0.002 / 0.004	0.009 / 0.011
3	0.349 / 0.408	0.623 / 0.674	0.785 / 0.788

more subtle, the same score / accuracy disparity is present and this is consistent over the train and test results. Of particular interest is the wide range of results returned in terms of score with consistently high values for results in overall accuracy. In the lower score quartile, a value of approximately 0.44 is returned while the upper quartile provides results above 0.7. In contrast, the overall accuracy results range from 0.933 to 0.935 between first and third quartiles, which again indicates a degree of degeneracy present in the solutions. This is confirmed by the class-wise results for detection and false positive rates, with a clear indication of preference for solutions strictly predicting the 92% majority class (3, or subnormal). Detection rate on all three quartiles for class 3 is near unity while false positive rates range from 0.4 to 0.78, with results being reasonably consistent over both train and test. Despite this indication of degeneracy, some highly specific learning appears to be taking place on class 1 with detection generalization being between .25 and .8 between the first and third quartiles, while false positive rates remain near zero. This is an interesting result as the training data for class 2 (hyperfunction) holds a 2:1 representational advantage over exemplars of class 1 (not hyperthyroid) on Thyroid, however this concept is clearly not being readily learned by the canonical GP.

Regardless of the modest success on class 1, the canonical GP results from E1 on Thyroid are further evidence of the class imbalance problems facing canonical multi-class GP classifiers. The inability of GP to sample an appropriate representation of each class naturally leads to guessing behavior representing a degenerate state (a local optima) that provides solutions that are not generally acceptable in the classification context. The CMGE system presented in this thesis employs balanced multi-class archiving of relevant training exemplars as evolution progresses and thereby directly enforces an appropriate sampling of the data from all class distributions. Comparative results will be provided in Section 5.2 that demonstrate improved robustness to the class imbalance problem, with the CMGE models providing significant improvements over the canonical results whilst greatly reducing training time and containing solution complexity over the large and unbalanced Census and Thyroid data sets. Moreover, the next natural modification to make to canonical GE will be to incorporate class balanced sub sampling (RssGE) where the two models will be benchmarked in Section

5.2.

5.1.2 Training Time

The widely acknowledged problem of scalability [6] [86], that is the ability to train the GP algorithm on large data sets, is frequently cited as a primary disadvantage of GP in comparison to other machine learning approaches. This is readily demonstrated for the canonical GP system by Experiment E1 over both the Census-Income and Thyroid data sets. Table 5.1 indicates the quartiles for GP training time in seconds under the 'Training time' heading. The median time to train the canonical GP on Census was 172,409 seconds (approximately 2,873 minutes, or roughly 48 hours). Only 25% of the time did the canonical framework train in under 39 hours, while another 25% of the initializations required at least 75 hours. As a result, the 20 independent initializations on Census required over a month to carry out on modern hardware, without the utility of parallel hardware. The Census data set contains approximately 200,000 training patterns and clearly demonstrates the inappropriateness of the traditional approach to training GP under a large-scale classification context.

Similarly, the considerably smaller Thyroid problem (3709 training exemplars) reflects the scalability pathology of canonical GP. Quartile training times, provided in table Table 5.2, ranged from 5241 to 8529 seconds (quartiles 1 and 3, respectively) with a median training time over 7000 seconds, or approximately 2 hours. The 50 independent initializations for E1 on the Thyroid problem required nearly a full week to complete, clearly demonstrating the prohibitive cost of the canonical GP approach to training on real-world classification tasks (in this three-class case, we naturally required construction of three separate classifiers).

Two factors contribute to the training pathology of canonical multi-class GP. The main impediment (in terms of training set size) is directly related to the requirement of the inner training loop: GP evaluates each individual in the population over the entirety of the training set to determine error as outlined in Section 2.5. The second issue is the requirement of separate initializations for each class. The degree to which this training overhead can be reduced by evaluation constraints, active learning and Pareto-coevoultionary frameworks will be addressed by the comparative results,

presented in Section 5.2. The CMGE system presented in this thesis will be shown to provide scalability by reducing the training time by orders of magnitude while containing solution complexity and simultaneously providing improved classification results across the current data sets.

5.1.3 Solution Complexity

Table 5.1 demonstrates the solution complexity of canonical GE over the Census problem under the 'Class-wise Soln Size' section. All solution sizes are provided as string lengths of the unsimplified phenotype expressions. Class 1 of the canonical Census solutions indicate more variability over the quartiles, ranging from 12 to 78.5 versus 13 to 67.5, despite a considerably smaller median solution size (27.5) than that of class 2 (41.5). This large degree of variation indicates the inability of canonical GP to contain code growth (or bloat), however this effect may be partly a result of utilizing only 20 runs to constrain the experiment times.

Solution sizes for the Thyroid problem under the canonical GP approach are provided in the solution size quartile summary of Table 5.2. Between the three classes the solution size results are approximately comparable over the three quartiles in terms of variation, with solution sizes ranging from the mid 20s to low 100s. The two classes demonstrating better detection rates (classes 1 and 3) having smaller median solution sizes. Of the two, class 1 solutions (providing modest median detection rates results whilst maintaining a low false positive rate) returned the smaller solution sizes over all quartiles; however, neither result is evidently different in terms of size or variation from the largely degenerate solutions of class 2. This result provides further evidence of canonical GP's inability to contain code growth during evolution. In the following sections, the CMGE framework presented in the current work will demonstrate that code growth can be contained by using explicit multi-objective optimization measures to apply evolutionary pressure for parsimony whilst learning classifiers that meet or improve upon classification results and training times returned by the canonical GP approach.

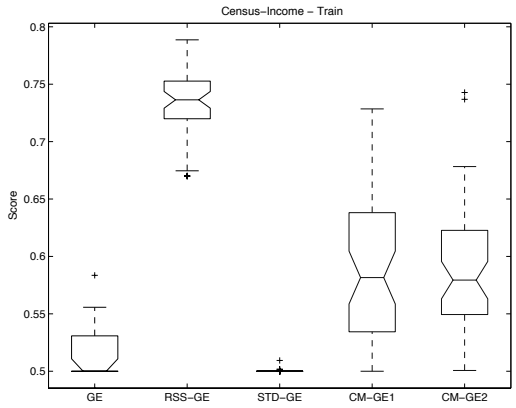
5.2 Comparisons: Canonical vs. Scalable GP

The results of Experiments E2-E5 (Table 4.6) will now be introduced in a comparison context with those of canonical GP. The degeneracy of canonical GP on both data sets in E1 (described in Section 5.1.1) will allow the overall accuracy results of classification performance to be ignored in the following comparisons in favor of the score metric, which indicates a balanced (class-wise) measure of overall classification performance. Results of classification performance, training time and solution complexity will demonstrate improvements due to the CMGE framework as well as the approximate equivalence of the canonical GE method with StdGE, the standard GE approach with runtime constrained by total number of evaluations that will be employed as a baseline indicator hereafter, Equation 4.1.

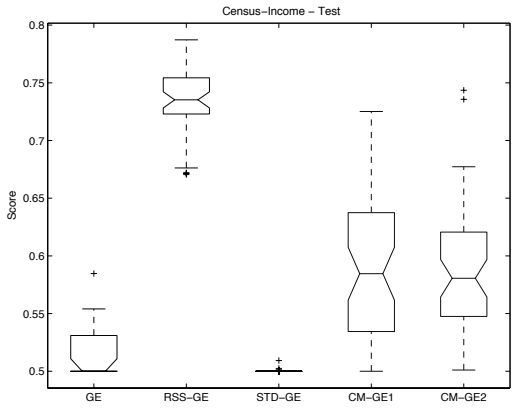
5.2.1 Classification Performance

Figure 5.1 provides graphical representations of train and test results of E1 (canonical GP) on Census as compared to E2-E5 results compiled for Census. The canonical results are indicated with the label GE in the box plots of Figure 5.1 (a) and (b) (train and test, respectively) and multiple comparison plots (c) (d) (e) and (f). Subplots (c) and (d) of Figure 5.1 provide multiple comparison of means based on ANOVA F-test results for train and test, respectively; while subplots (e) and (f) provide multiple comparison results based on the Kruskal Wallis non-parametric version of ANOVA (see Section 4.4.4) of train and test, respectively. As there is limited evidence that the assumptions of the classical ANOVA F-test are valid (particularly those requiring performance results having equal variance and approximate normality), statistical significance will be indicated by an agreement in results of the two multiple comparison analysis models. Figure 5.2 provides similar classification performance comparisons between E1 and E2-E5 on the Thyroid data set.

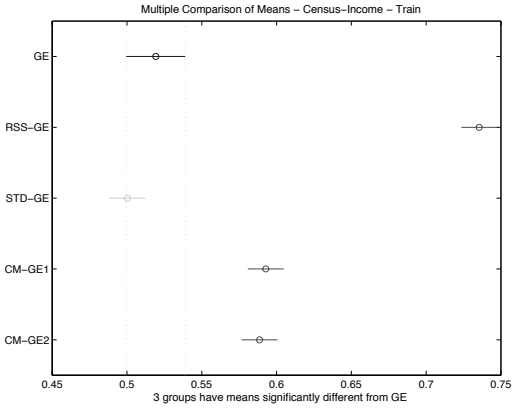
The Census score results of Figure 5.1, subplots (a) and (b) indicate that the low median score returned by canonical GP (indicated simply by GE) is, in fact, the worst in terms of classification performance of all GE systems examined in this study, with StdGE providing a very similar result with less variation (near zero) over the quartiles. These results are extremely consistent between train and test,



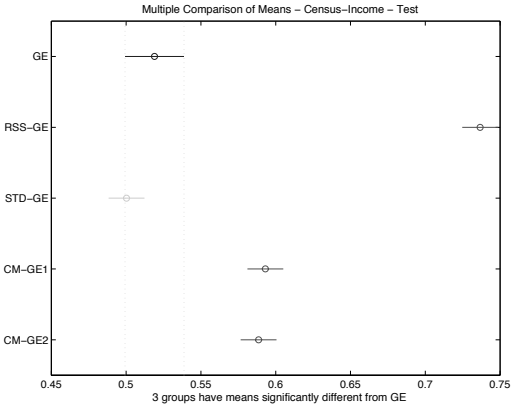
(a) Score (Train)



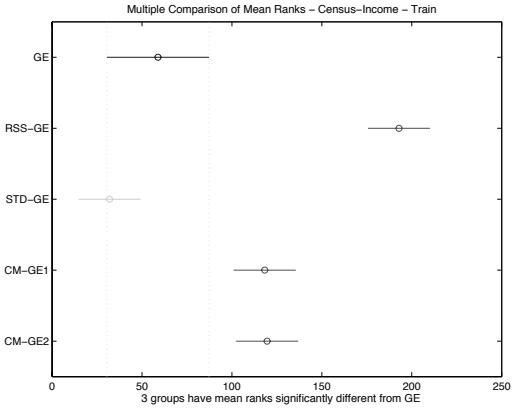
(b) Score (Test)



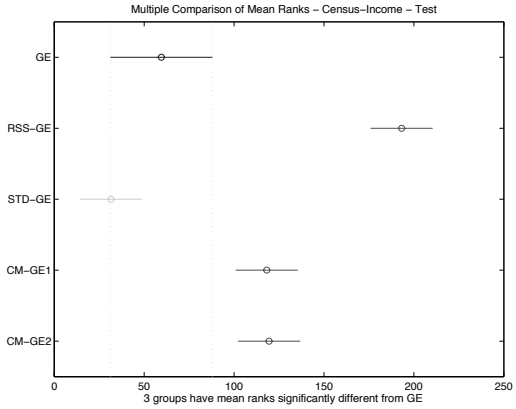
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



(e) Comparison of Mean Ranks (Train)

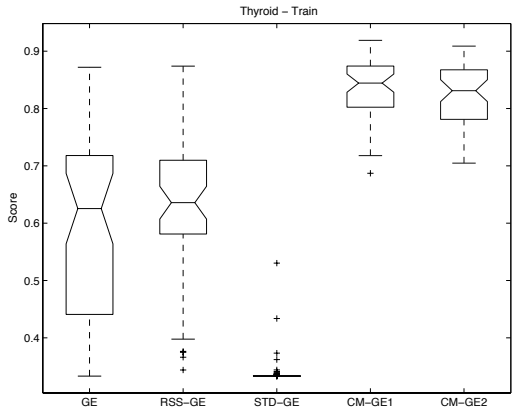


(f) Comparison of Mean Ranks (Test)

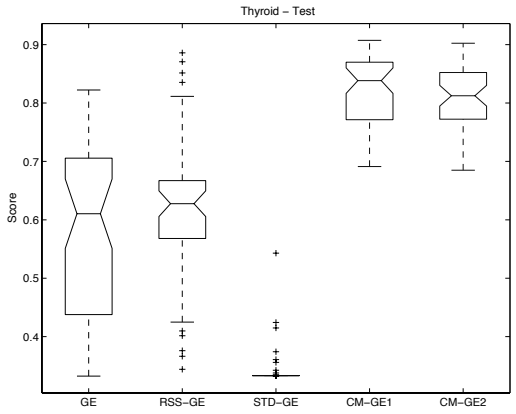
Figure 5.1: Direct comparison of Canonical GP (GE) performance on CENS.

indicating that both systems are providing essentially degenerate results on this highly unbalanced data set. Moreover, the score results of StdGE and canonical GP are shown to be the same, from the perspective of statistical significance; both the F-Test and Kruskal Wallis multiple comparison plots demonstrate that only RssGE and the two CMGE models returned score results that were improved with statistical significance. Of the improved results, RssGE appears to return the best score results with the least variation over the quartiles, however both CMGE1 and CMGE2 provide meaningful improvements over the standard models and both return maximum results that were in the same range as those of the RssGE model. These results are consistent between train and test, indicating that the three models returning improved results generalized readily on the Census data. Of the CMGE systems, CMGE2 was more consistent in terms of score and returned better maximum score, however there is no statistical significance in the difference between these results according to the multiple comparison results of Figure 5.1 subplots (c) - (f).

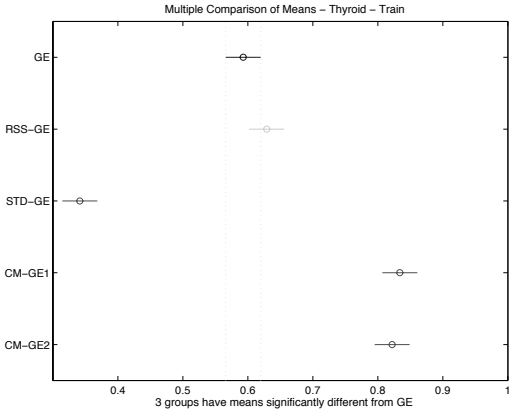
Thyroid score results of Figure 5.2, subplots (a) and (b) again indicate poor results for the standard models (canonical GP and StdGE), however there is a significant preference for the canonical model over the limited evaluation StdGE. In both train and test results the basic models provide the lowest median scores and canonical GP exhibits the most variation with quartile scores ranging from values of .45 to approximately 0.70, however the maximum result on test does not exceed 0.85. StdGE consistently returns degenerate solutions, with some outlying points in both train and test which indicates only a modest number of cases where learning extends beyond one class. RssGE, while not statistically improving on canonical GP, does return better median and maximum scores on test than either StdGE or canonical GP, indicating better generalization in the high-end results. The CMGE approaches are not significantly different from one and other however provide significant and appreciable improvements over all other models. In terms of consistency, both CMGE1 and CMGE2 demonstrate marked improvements over canonical GE in terms of interquartile range.



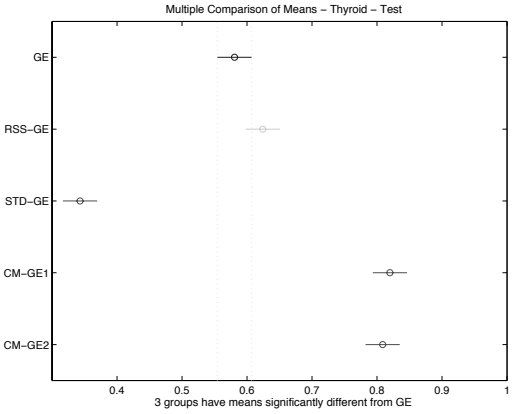
(a) Score (Train)



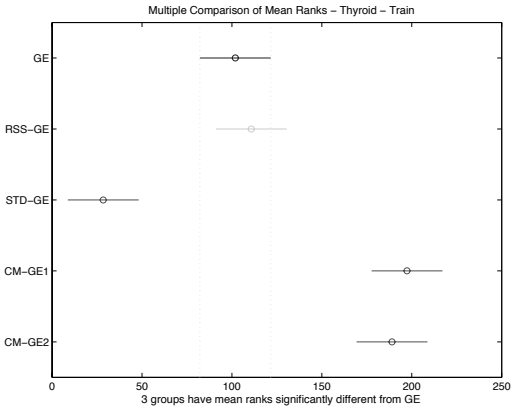
(b) Score (Test)



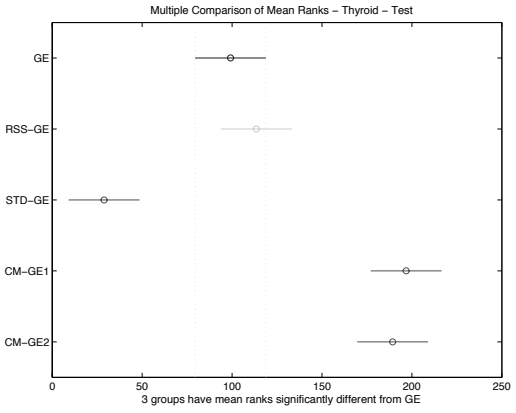
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

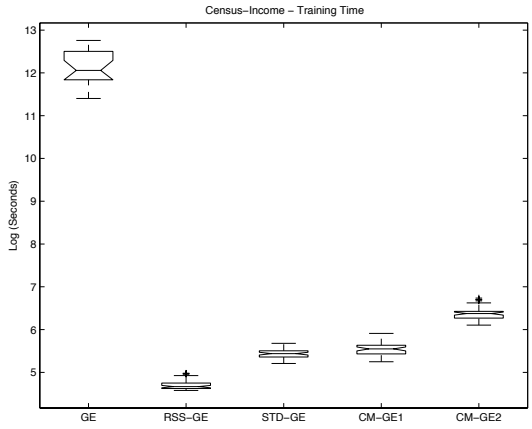


(e) Comparison of Mean Ranks (Train)

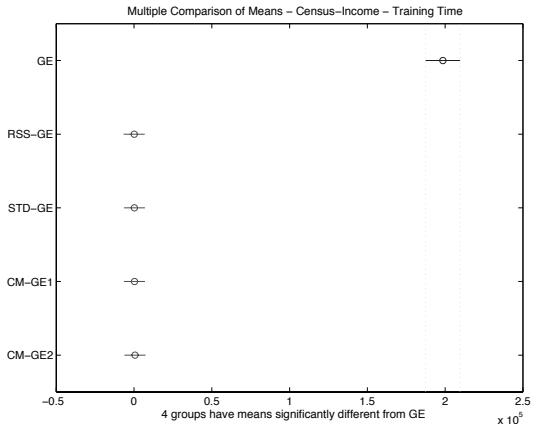


(f) Comparison of Mean Ranks (Test)

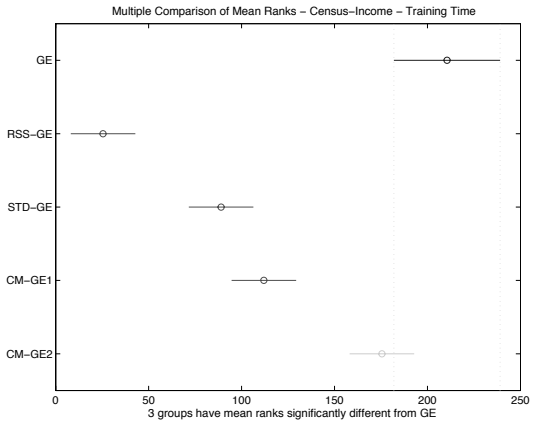
Figure 5.2: Direct comparison of Canonical GP (GE) performance on THYD.



(a) Training Time



(b) Comparison of Means



(c) Comparison of Mean Ranks

Figure 5.3: Direct (GE) comparison of training time on CENS.

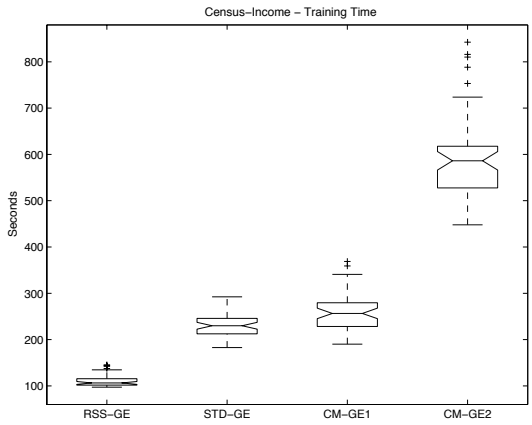
5.2.2 Training Time

Comparative results for training time on the Census data set are presented in Figure 5.3. The y-axis of subplot (a) is a log-scale of time in seconds. This result indicates a clear and substantial difference in the training times between the canonical and scalable GP approaches, where there is a difference of four orders of magnitude. The multiple comparison of means (subplot (b)) bears out this difference indicating statistical significance at the 95% confidence level. The results of the four scalable approaches are isolated for comparison in Figure 5.4, where CMGE2 returns the largest median training time with quartiles ranging from approximately 500-600 seconds; CMGE1 and StdGE return similar medians with quartiles ranging from 200-300 seconds, and RssGE the lowest median at just over 100 seconds, with very little variation in this result. While there appears to be a significant difference in training times between these methods, the practical significance of minutes is debatable when significant performance gains are attainable.

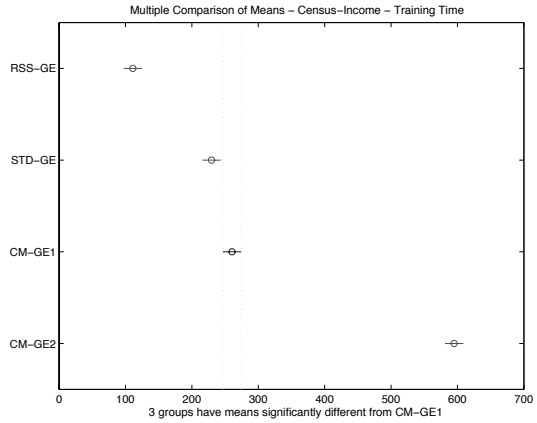
A similar comparison outcome is demonstrated by the results for training time on the Thyroid data set in Figure 5.5, where the log scale of seconds is again used on the y-axis of subplot (a). This result indicates a difference of an order of magnitude between canonical GP and the scalable models, with the multiple comparisons of subplots (b) and (c) confirming that this is a statistically significant result. Of the scalable results (presented in isolation on a linear time scale in Figure 5.6), CMGE2 again returns the greatest median training time of just over 100 seconds, followed by CMGE1 at around 1 minute; StdGE and RssGE both returned median training times of under a minute. The difference between the fastest and slowest training times among the scalable GP models is thus approximately one minute.

5.2.3 Solution Complexity

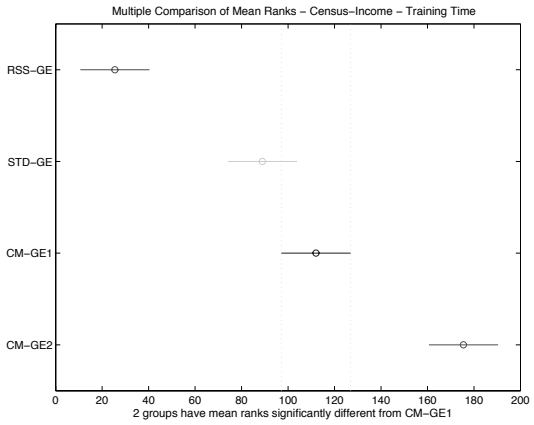
Solution complexity comparisons (as measured by expression string length) for the Census and Thyroid Data sets are presented in Figures 5.7 and 5.8, respectively. In the case Census results, the canonical form of GP returned solution complexities with the most variability as well as largest solutions according to the upper outlier and third quartile results of subplot (a); these results were followed closely by StdGE,



(a) Training Time

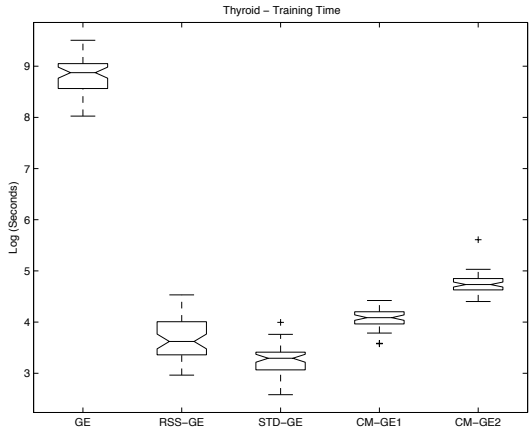


(b) Comparison of Means

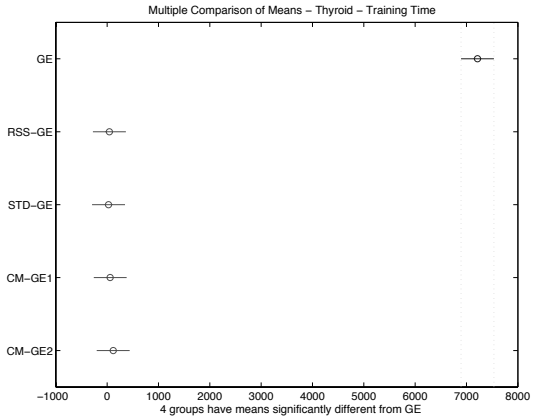


(c) Comparison of Mean Ranks

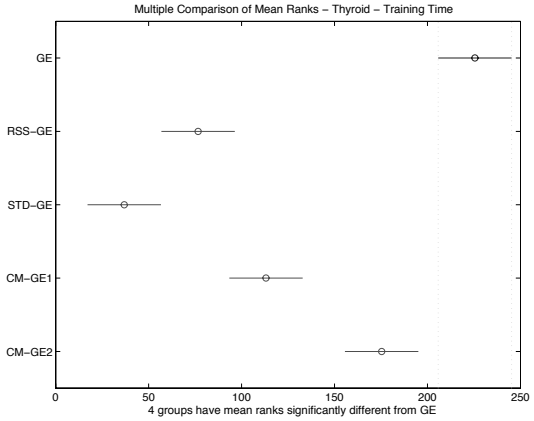
Figure 5.4: Direct (GE) comparison of training time on CENS.



(a) Training Time

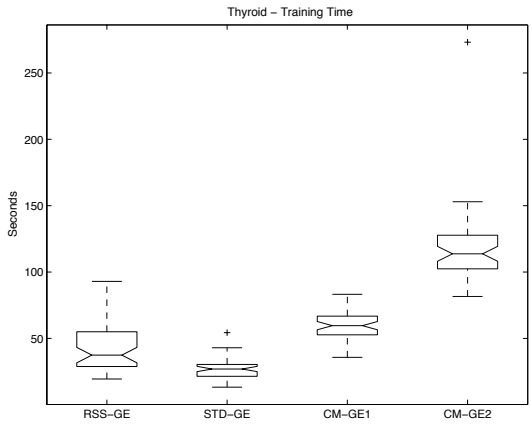


(b) Comparison of Means

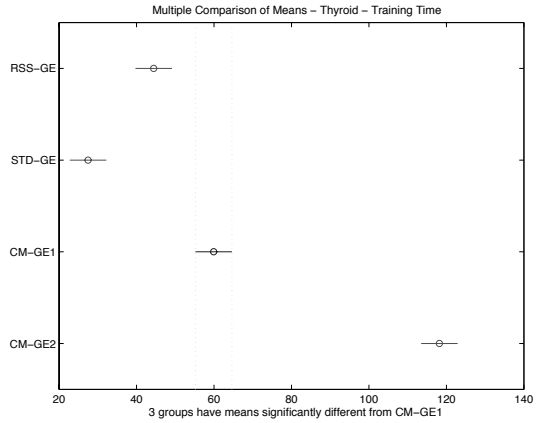


(c) Comparison of Mean Ranks

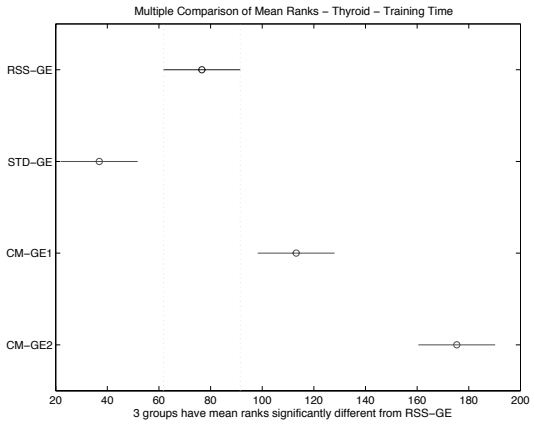
Figure 5.5: Direct (GE) comparison of training time on THYD.



(a) Training Time

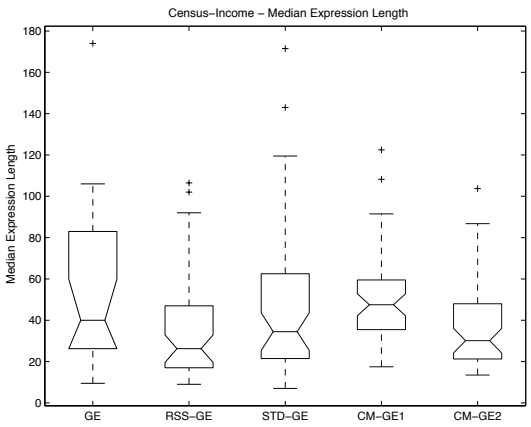


(b) Comparison of Means

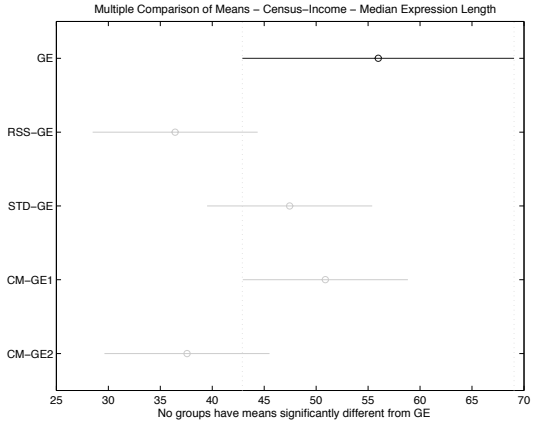


(c) Comparison of Mean Ranks

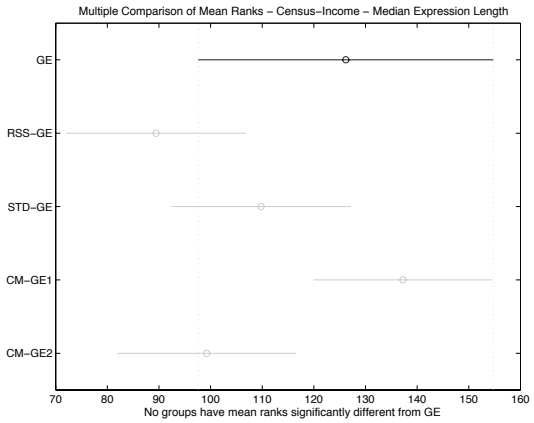
Figure 5.6: Direct (GE) comparison of training time on THYD.



(a) Solution Length (strlen)



(b) Comparison of Means



(c) Comparison of Mean Ranks

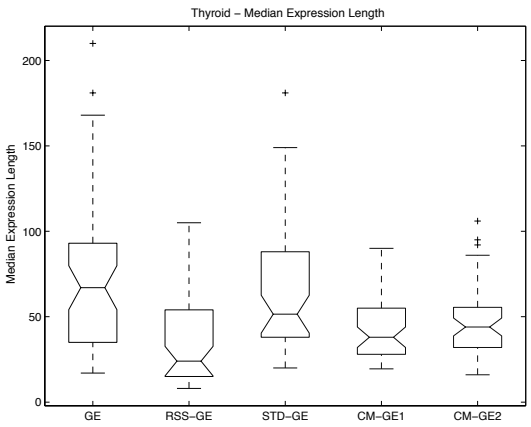
Figure 5.7: Direct (GE) comparison of solution length on CENS.

which also returned the greatest maximum solution complexity of any approach. Of the three remaining models, RssGE had the lowest median complexity, followed closely by CMGE2; however, according to the multiple comparisons of subplots (b) and (c), none of these results carry statistical signifance. Both of the competitive models were able contain the variability in solution complexity better than any of the other approaches.

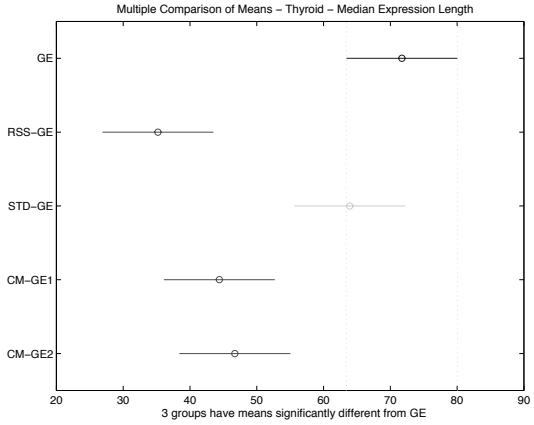
Comparative results for Thyroid solution complexity (Table 5.8) illustrate the tendency toward solution growth in the standard models, with the canonical and StdGE models returning the largest complexity results across all quartiles (subplot (a)), with subplots (b) and (c) confirming the statistical significance of these results. Moreover, the variability in the canonical and StdGE results is appreciably greater than any of RssGE or the CMGE models. RssGE returned the lowest first and second (median) quartile results however, these groups were not shown to be statistically different by the multiple comparison plots in subplots (b) and (c). Similarly to the Census results, both of the CMGE results demonstrated less variation in solution complexity than any of the other approaches.

5.3 Scalable Framework Comparisons

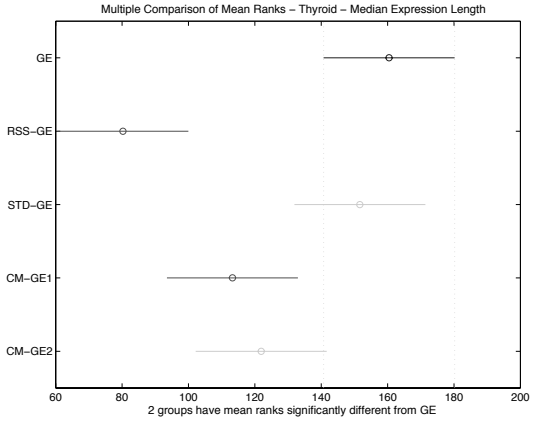
Experiments E2-E5 provide the basis for comparisons across the scalable GP models (RssGE, StdGE, and the competitive multi-objective models CMGE1 and CMGE2) over 12 classification problems. Results will be discussed in the following sections, using summary tables indicating performance rankings on overall classification (in terms of accuracy and score on both train and test data), training time and solution complexity. Bold rankings under the CMGE headings denote statistically significant improvements over the best performing model between RssGE or StdGE according to both F-Test and Kruskal Wallis multiple comparison plots. Similarly, bold rankings under the RssGE or StdGE headings represent statistically significant improvements from the best performing CMGE model. Underlined rankings indicate results that are statistically significant improvements over all competing GP models. The lower portion of each rank summary table provides performance summaries, in terms of sum of ranks (Total), average rank (Avg.), standard deviation of rank (Std.), number of best



(a) Solution Length (strlen)



(b) Comparison of Means



(c) Comparison of Mean Ranks

Figure 5.8: Direct (GE) comparison of solution length on THYD.

rankings (N_1), number of worst ranks (N_4), number of bold ranks (**N**) and number of underlined ranks (N). Where appropriate, quartile summaries, box plots and multiple comparison charts will be provided in addition to the rank summary tables in order to highlight representative (or atypical) results in detail on each performance metric. Full scalable GP quartile summary listings are provided in Appendix G, Sections G.3 and G.2. Scalable GP comparison results for each model / data set combination are provided in Appendix A, Section A.2.

5.3.1 Classification Performance

Table 5.3: Comparison of scalable GP accuracy ranks (train)

Data Set	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
BOST	3	4	**2**	**1**
BUPA	4	3	**2**	**1**
CENS	4	1	3	2
CONT	4	3	**2**	**1**
IMAG	4	3	**1**	**2**
IRIS	4	3	**2**	**1**
KD99	3	4	**2**	**1**
PIMA	4	3	**1**	**2**
SHUT	4	3	2	**1**
THYD	4	3	2	1
WINE	4	3	**1**	**2**
WISC	4	3	2	**1**
Stat	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
Total	46	36	22	16
Avg.	3.83	3	1.83	1.33
Std.	0.39	0.74	0.58	0.49
N_1	0	1	3	8
N_4	10	2	0	0
N	0	0	8	10
N	0	0	0	1

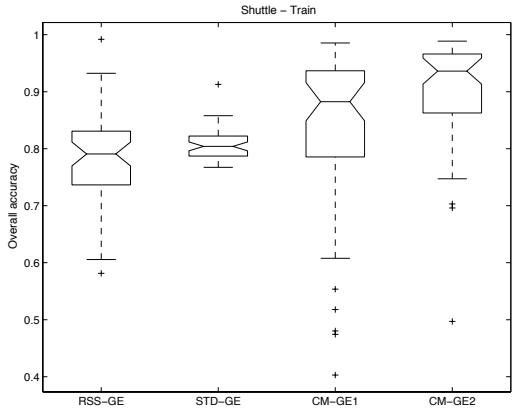
Table 5.3 lists rankings of median overall accuracy results on training data for the four scalable GP models. Over the 12 data sets, the CMGE models return median training accuracies that claim the top two rankings 23 out of 24 times with 18 of these results representing significantly better training accuracies than either of the StdGE or RssGE baselines. Of the CMGE models, the two-stage pruning CMGE2 provides the best training accuracy results, being top ranked 8 times, improving on the baselines 10 times and once returning results significantly better than the competing frameworks. CMGE2 thus had the best average rank of 1.33 and standard deviation

of 0.49, while the worst results were returned by RssGE, which was ranked last 10 of 12 times. The StdGE framework returned results that appeared to moderately improve on RssGE in training, however many of these training results (including the top ranked training accuracy returned on CENS) will be shown to have produced degenerate solutions.

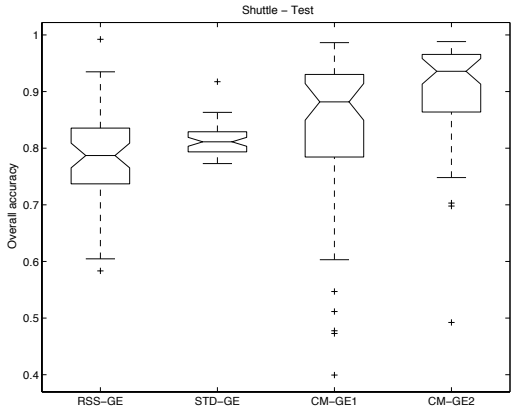
Table 5.4: Comparison of scalable GP accuracy ranks (test)

Data Set	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
BOST	3	4	**1**	**2**
BUPA	4	3	**2**	**1**
CENS	4	1	3	2
CONT	4	3	**2**	**1**
IMAG	4	3	**1**	**2**
IRIS	4	3	**1**	2
KD99	3	4	**1**	**2**
PIMA	4	3	**1**	**2**
SHUT	4	3	2	**1**
THYD	4	1	3	2
WINE	4	3	**1**	**2**
WISC	4	3	**2**	**1**
Stat	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
Total	46	34	20	20
Avg.	3.83	2.83	1.67	1.67
Std.	0.39	0.94	0.78	0.49
N_1	0	2	6	4
N_4	10	2	0	0
N	0	0	9	10
N	0	0	1	2

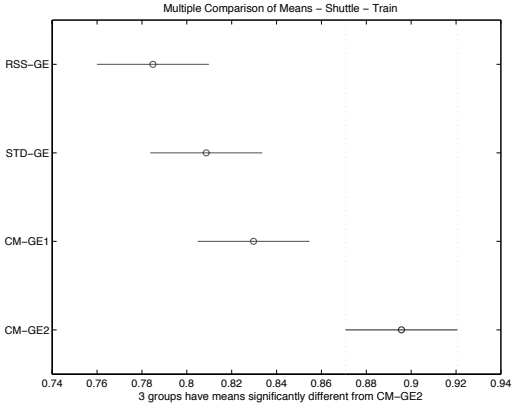
Median test accuracy rankings for the scalable GP models are provided in Table 5.4. The best generalization performance according to accuracy was associated with the CMGE frameworks, with 22 of the top 24 ranks being occupied by either CMGE1 or CMGE2. Of these results, 19 represent significant improvements over the baseline approaches. Between the two CMGE models, CMGE1 (despite lower training results) may have tended to generalize slightly better than CMGE2, with 6 top ranked median test accuracies; however both CMGE systems have the same average median test ranking, with CMGE2 showing less variation in this statistic (being ranked either 1 or 2 on every data set). Both the CENS and THYD problems resulted in CMGE1 having a rank of 3 and StdGE being ranked 1; however, neither of these results are representative of the true generalization performance of StdGE because of the widely unbalanced class distributions that allow degenerate solutions (that return the largest



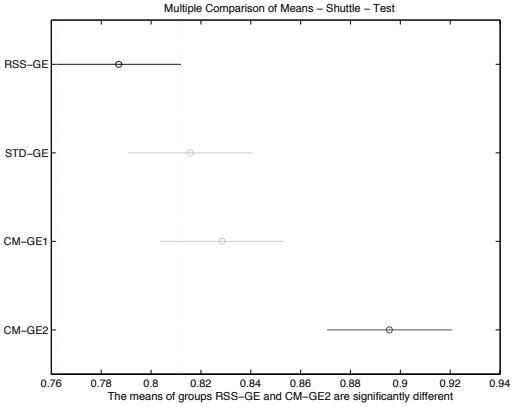
(a) Accuracy (Train)



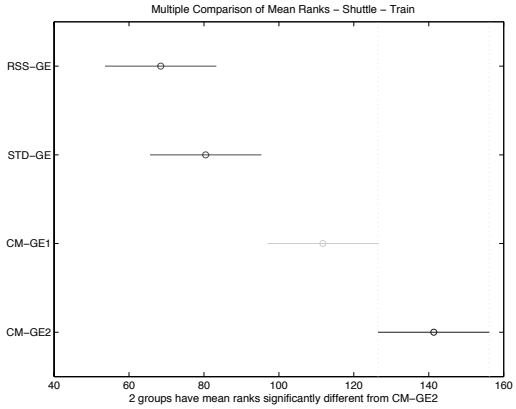
(b) Accuracy (Test)



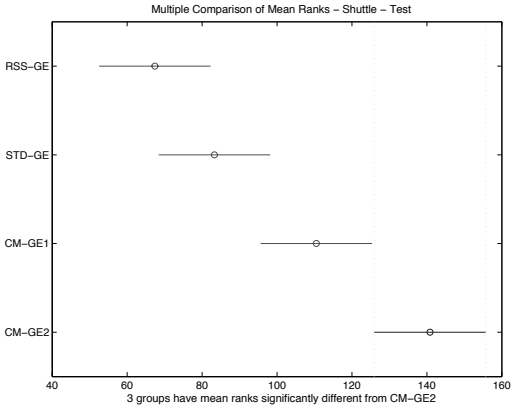
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



(e) Comparison of Mean Ranks (Train)



(f) Comparison of Mean Ranks (Test)

Figure 5.9: Direct comparison of scalable GP (GE) performance on SHUT.

class by default) to receive a high accuracy. An example of the degenerate behavior in StdGE (on THYD) was provided in Figure 5.2 and discussed along with the canonical comparative results in Section 5.2.1. Figure 5.9 (train / test accuracy on the Shuttle data set) provides a typical example of the comparative accuracy results over the four scalable models in detail. Subplots (a) and (b) demonstrate that, despite considerably more variation in accuracy on both train and test respectively, the CMGE frameworks return median accuracies that are in the mid 0.8 to mid 0.9 range, with both baseline medians being less than 0.85. Moreover, the CMGE2 model's lower quartile results are considerably better than the upper quartiles of either baseline approach and near the median CMGE1 result. Subplots (d) and (f) demonstrate the statistical significance of the CMGE2 results over all competing models in test, resulting the underlined rank of 1 in the SHUT row, CMGE2 column of Table 5.4. The results are consistent over Shuttle train and test data, however CMGE2 did not demonstrate significant improvement according to the Kruskal Wallis multiple comparison on training data (subplot (e)) against CMGE1.

Table 5.5: Comparison of scalable GP score ranks (train)

Data Set	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
BOST	3	4	**2**	**1**
BUPA	3	4	2	**1**
CENS	**1**	4	2	3
CONT	**1**	4	2	3
IMAG	4	3	**1**	**2**
IRIS	4	3	**2**	**1**
KD99	3	4	**1**	**2**
PIMA	3	4	**2**	**1**
SHUT	3	4	**1**	**2**
THYD	3	4	**1**	**2**
WINE	4	3	**1**	**2**
WISC	4	3	**2**	**1**
Stat	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
Total	36	44	19	21
Avg.	3	3.67	1.58	1.75
Std.	1.04	0.49	0.51	0.75
N_1	2	0	5	5
N_4	4	8	0	0
N	2	0	9	10
N	2	0	0	3

Median score rank results are presented in Tables 5.5 and 5.6 for training and test, respectively. In the training results of Table 5.5, 22 of the top 24 ranks are

Table 5.6: Comparison of scalable GP score ranks (test)

Data Set	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
BOST	3	4	**1**	**2**
BUPA	3	4	2	**1**
CENS	**1**	4	2	3
CONT	**1**	4	2	3
IMAG	4	3	**1**	**2**
IRIS	4	3	**1**	**2**
KD99	3	4	**1**	2
PIMA	3	4	**2**	**1**
SHUT	3	4	**2**	**1**
THYD	3	4	**1**	**2**
WINE	4	3	**1**	**2**
WISC	4	3	**2**	**1**
Stat	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
Total	36	44	18	22
Avg.	3	3.67	1.5	1.83
Std.	1.04	0.49	0.52	0.72
N_1	2	0	6	4
N_4	4	8	0	0
N	2	0	9	9
1	2	0	1	2

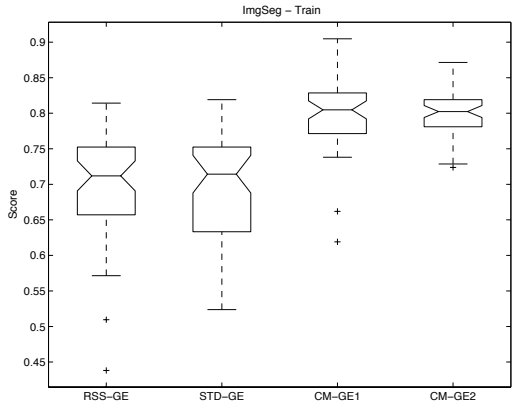
occupied by the CMGE models with 19 of these being supported by statistically significant improvements over the two baselines, RssGE and StdGE. CMGE2 returned training scores that were statistically significant improvements over all of the competing models on three occasions (the BUPA, PIMA and WISC problems), and each of the CMGE systems had the top ranking 5 times. The CMGE approaches tended to trade off the top ranking between data sets, however twice neither was able to improve on RssGE, these being the CENS and CONT problems where RssGE results were top ranked and significant improvements over all competing approaches. Of the score training results, StdGE tended to acquire the worst ranking the most frequently (8 times in total) and the third ranking on 4 occasions, where 3 of these results corresponded to relatively easy problems including IRIS, WINE, and WISC. We attribute this to all three data sets being unbalanced. That is to say, although IRIS and WINE have an equal number of exemplars per class, the multi-class nature of the data sets renders them 'unbalanced' from the binary classification context. In the case of WISC, the problem is binary, however with an unequal class distribution. The best results in terms of training score were clearly CMGE1 and CMGE2, with a slight preference for CMGE1, having the lowest average rank (1.58) and the least

amount of variation in rank (0.51) as it was always assigned a rank of either 1 or 2.

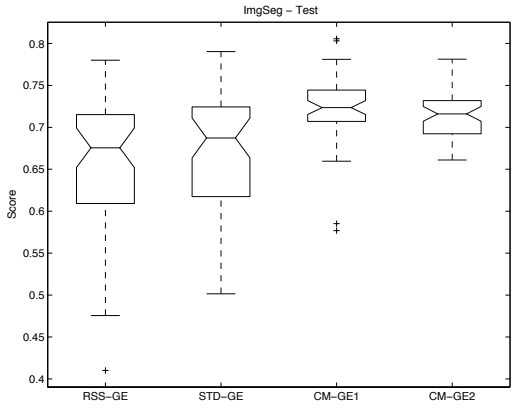
Test score rankings generally confirm the results of accuracy with both the CMGE models able to generalize better than either RssGE or StdGE on 22 of 24 occasions. CMGE1 always received one of the top two rankings, resulting in the best average rank (1.5) and lowest variation (0.52). In all systems, average rankings and standard deviation of rank on test score statistics are very consistent with the score performances on training data. CMGE2 was again able to perform significantly better than any other competing model on BUPA and WISC while still maintaining the top ranking on PIMA from the training results; however, it was CMGE1 that had the most top rankings (6) and tied with CMGE2 with 9 rankings that carried improvements that were statistically significant over the baseline models. On all but the easiest data sets (including IRIS, WINE and WISC) and IMAG, StdGE received the lowest rank, resulting in the worst average rank (3.67) with very low standard deviation (0.49). Figures 5.10 and 5.11 are indicative of the typical comparisons between the scalable systems in terms of of train and test score differences. Figure 5.10, subplots (a) and (b) illustrate the small margins of improvement in median score that StdGE had over RssGE in training and test. The CMGE models were typically much better in terms of score on both train and test while being better able to contain the variation in results. Moreover, the larger data sets (e.g., KD99 of Figure 5.11 subplots (a) and (b) for train and test, respectively) typically had larger margins in performance between baseline and CMGE, while the smaller or easier data sets typically returned modest improvements in terms of score in favor of CMGE1 and CMGE2.

5.3.2 Training Time

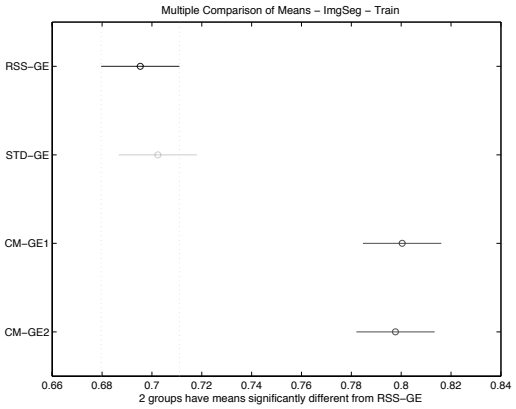
Table 5.7 provides the results of the four scalable GP median training time ranks. On all but one of the data sets, both RssGE and StdGE were able to train significantly faster than the CMGE models; however, the practical difference in terms of training time was typically negligible (seconds or a few minutes) when compared with the times that would be required for an unconstrained run of canonical GP (hours or days on large data sets). Certain CMGE results, specifically those returned by CMGE2, were as much as several times slower than the limited RssGE or StdGE training



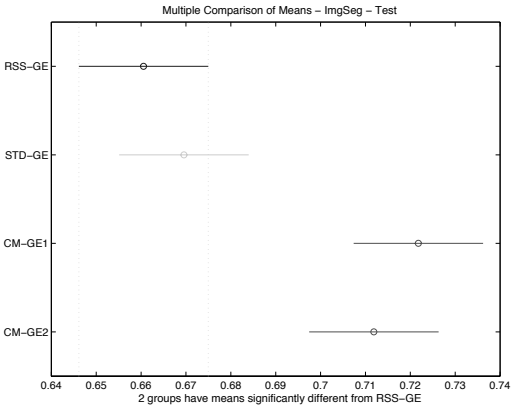
(a) Score (Train)



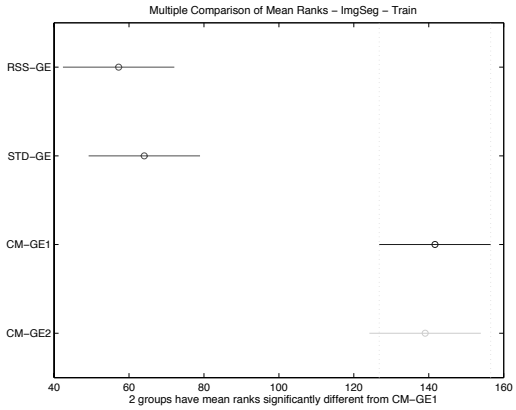
(b) Score (Test)



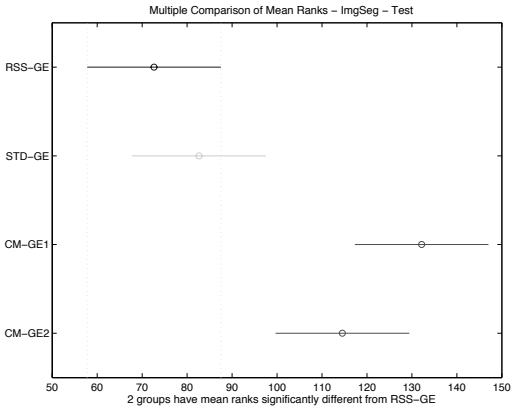
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

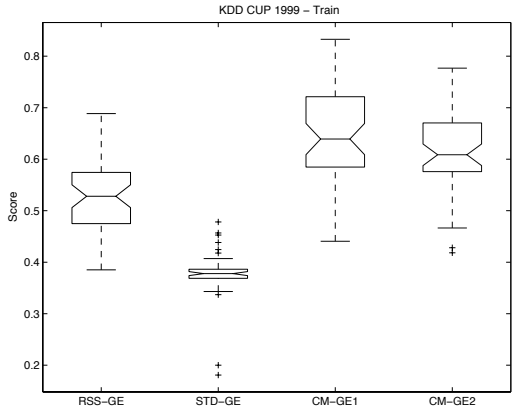


(e) Comparison of Mean Ranks (Train)

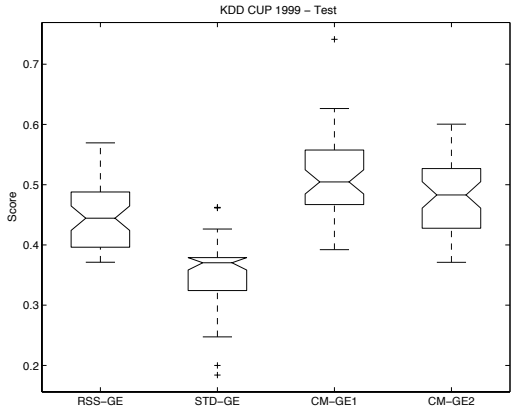


(f) Comparison of Mean Ranks (Test)

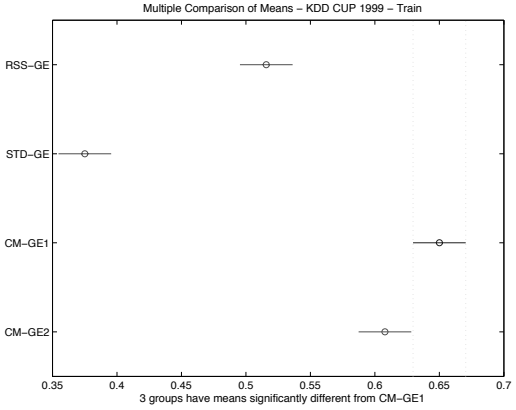
Figure 5.10: Direct comparison of scalable GP (GE) performance on IMAG.



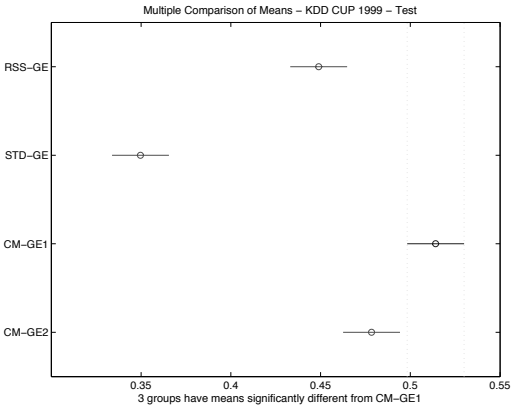
(a) Score (Train)



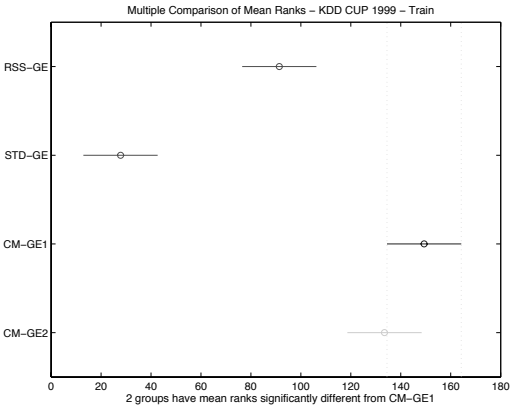
(b) Score (Test)



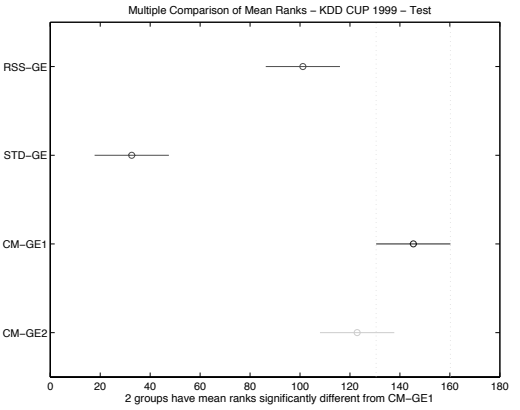
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

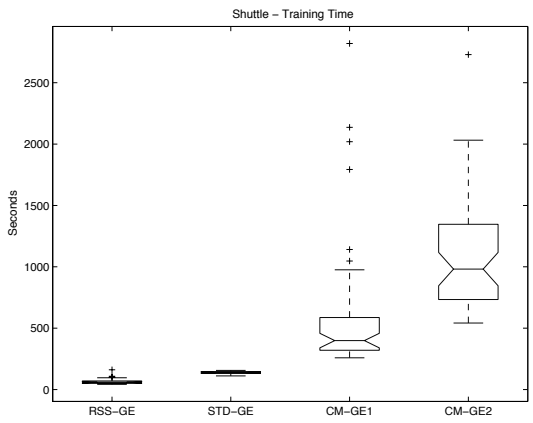


(e) Comparison of Mean Ranks (Train)

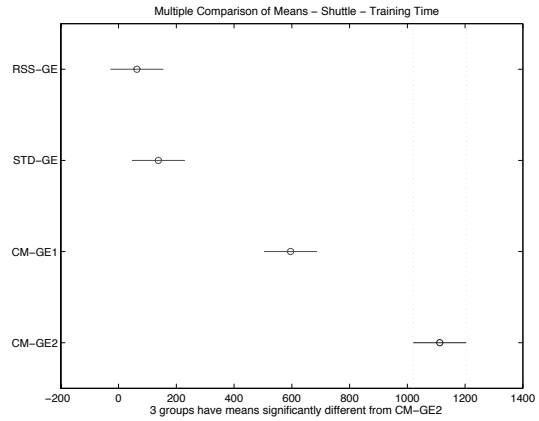


(f) Comparison of Mean Ranks (Test)

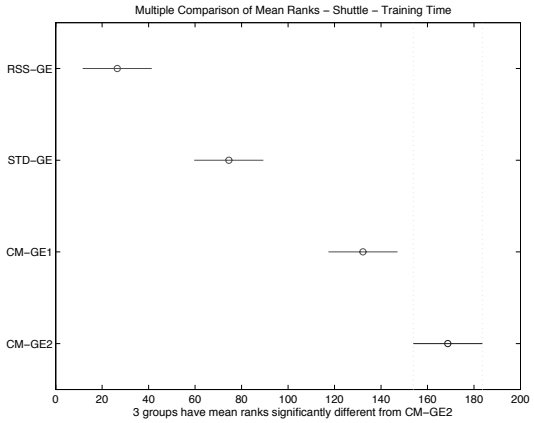
Figure 5.11: Direct comparison of scalable GP (GE) performance on KD99.



(a) Training Time



(b) Comparison of Means



(c) Comparison of Mean Ranks

Figure 5.12: Direct (GE) comparison of training time on SHUT.

Table 5.7: Comparison of scalable GP training time ranks

Data Set	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
BOST	**1**	**2**	3	4
BUPA	**2**	**1**	3	4
CENS	**1**	2	3	4
CONT	**2**	**1**	3	4
IMAG	**2**	**1**	3	4
IRIS	**2**	**1**	3	4
KD99	**1**	**2**	3	4
PIMA	**2**	**1**	4	3
SHUT	**1**	**2**	3	4
THYD	**2**	**1**	3	4
WINE	**2**	**1**	3	4
WISC	**2**	**1**	3	4
Stat	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
Total	20	16	37	47
Avg.	1.67	1.33	3.08	3.92
Std.	0.49	0.49	0.29	0.29
N_1	4	8	0	0
N_4	0	0	1	11
N	12	11	0	0
$\underline{N}$	3	6	0	0

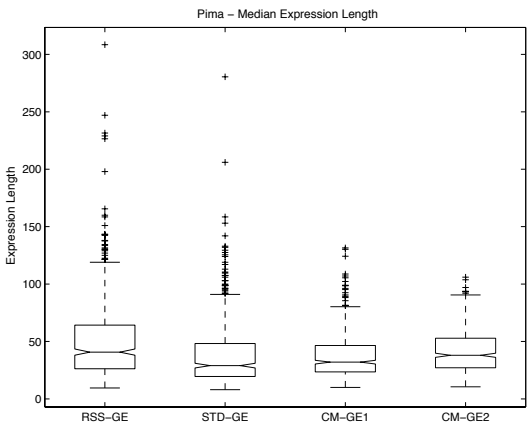
times. The typical differences will be illustrated with timing box plots; however, in the majority of cases training times for the CMGE models did not exceeding several minutes per initialization, well within tolerable bounds for most learning applications in contrast to time typically required to train canonical GP. Between the CMGE models, CMGE1 (employing a simple pruning algorithm) always ranked better than CMGE2 (aside from the PIMA data set where the ranks were reversed) and on 6 of the data sets there was a statistically significant difference in favor of CMGE1. Figure 5.6 illustrates the comparative training times on THYD, discussed earlier in Section 5.2.2 and is indicative of a typical difference in medians (approximately one minute) and a moderate difference was demonstrated by the box plot results for CENS in Figure 5.4. Perhaps the largest difference across all data sets in the current results is represented by the Shuttle results of Figure 5.12 where both CMGE1 and CMGE2 required hundreds of seconds to train. Naturally the major contributing factor to the training overhead of CMGE relative to RssGE and StdGE is the addition of the clustering algorithm and the multiple objective evaluations; moreover the runtime of the clustering algorithm is problem dependent (i.e., a function of the number of clusters identified).

5.3.3 Solution Complexity

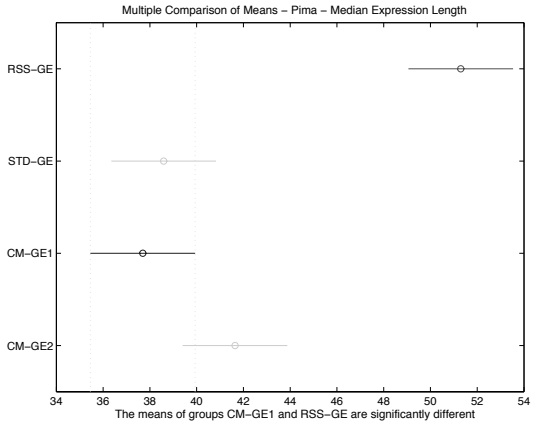
Table 5.8: Comparison of scalable GP solution length ranks

Data	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
BOST	3	4	2	1
BUPA	2	1	3	4
CENS	1	3	4	2
CONT	3	1	4	2
IMAG	4	3	2	**1**
IRIS	4	3	**2**	**1**
KD99	3	1	4	2
PIMA	4	1	3	2
SHUT	3	2	4	1
THYD	1	4	2	3
WINE	3	4	**2**	**1**
WISC	4	3	2	**1**
Stat	**Rss-GE**	**Std-GE**	**CMGE1**	**CMGE2**
Total	35	30	34	21
Avg.	2.92	2.5	2.83	1.75
Std.	1.08	1.24	0.94	0.97
N_1	2	4	0	6
N_4	4	3	4	1
N	0	0	2	4
N	0	0	0	2

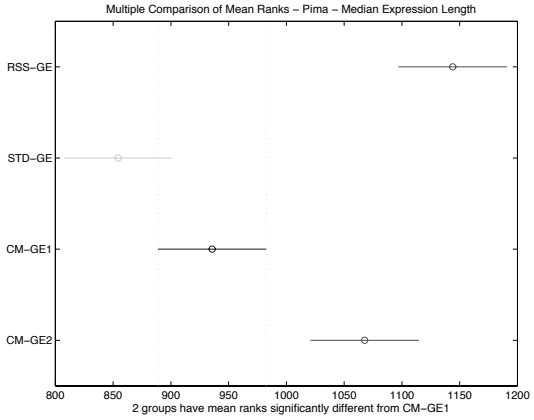
Median solution complexity ranks are provided in Table 5.8. Sixteen of the top 24 ranks are assigned to the CMGE models, with CMGE2 being top ranked (having smallest median solution size) on half of all data sets, 4 of these indicating statistical improvements over RssGE and StdGE results. On IMAG and WISC, the CMGE2 results were improvements that carried statistical significance over all other models. With an average rank of 1.75 and standard deviation of 0.97, CMGE2 outperforms CMGE1 (average rank of 2.83 with approximately the same variability in ranks). The worst results in terms of complexity ranks is RssGE, with a mean rank of 2.92; however StdGE returns the most variation in rank at 1.24. No baseline models were able to show a statistically significant improvement over the CMGE results, despite the fact that code growth should be at a minimum in these systems employing limited evaluations. Typical comparisons in terms of actual quartiles of string length are provided by Figures 5.13 (PIMA) and 5.14 (IRIS). PIMA solution length comparison results (subplot (a)) are representative of the cases where CMGE models return short expressions with well contained variation (indicated by quartiles) but do not significantly improve on both baselines as indicated by the multiple comparison results



(a) Solution Length (strlen)

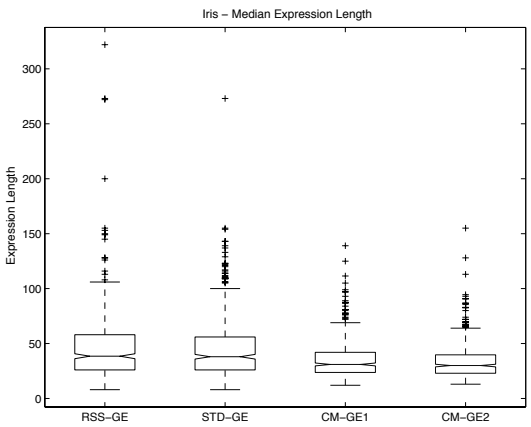


(b) Comparison of Means

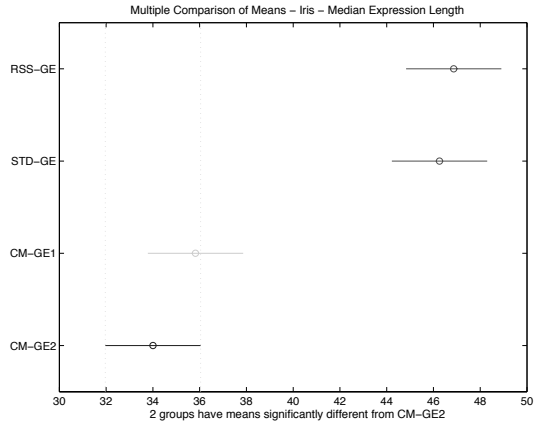


(c) Comparison of Mean Ranks

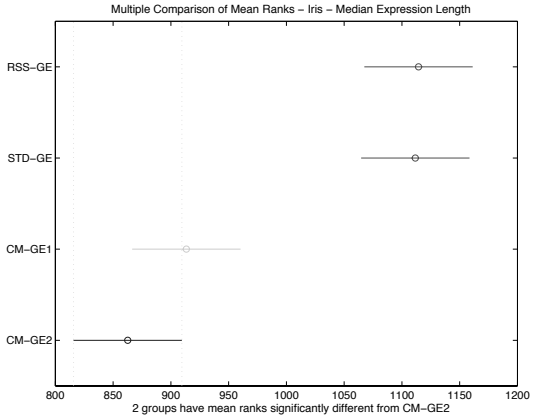
Figure 5.13: Direct (GE) comparison of solution length on PIMA.



(a) Solution Length (strlen)



(b) Comparison of Means



(c) Comparison of Mean Ranks

Figure 5.14: Direct (GE) comparison of solution length on IRIS.

in subplots (b) and (c). The IRIS results of Figure 5.14 indicate a typical scenario for the cases where classification performance is near perfect over all classifiers (e.g., IRIS, WINE, WISC). In these situations both of the CMGE frameworks are able to provide equal or better classification performance while finding solutions that are significantly smaller than the baseline approaches, as demonstrated by the multiple comparison results of Figure 5.14 subplots (b) and (c).

5.4 Concerning Multi-class PGEC Results

A complete analysis of multi-class PGEC results is omitted as our variant of the PGEC algorithm (configured as a multi-initialization classifier) returns multi-class solutions that are largely degenerate and not competitive with the baseline models. This is readily confirmed by the overall and class-wise quartile result summaries supplied in Appendix G, where PGEC typically returns the lowest classification performance independent of the data set. Basic observation of PGEC runs appears to indicate that the model does not perform well in a multi-class configuration on account of inconsistency or incompatibility between initializations corresponding to the various classes. For example, a typical multi-class run might find a good (majority voting) team for one specific class – i.e., strong intra-class behavior, having high detection rate with a low false positive rate; however, the teams evolved on the separate initializations for the remaining classes can be so poor (or have such detrimental effects on the end results) that the performance of the final solutions can be highly unreliable. In other words, the team models rely on strong intra-class behavior as well as strong inter-class behavior to achieve acceptable performance. In the case of PGEC, teams (corresponding to classes) are evolved independently without consideration for the multi-class voting context and therefore the latter behavior might occur only by chance. Specifically, PGEC's fragile inter-class behavior can occur when teams from separate initializations are of considerably different sizes (i.e., large class-wise variation in team size) are finally employed in the multi-class, majority vote setting. In such a scenario, large teams of learners can be more likely to 'win' majority votes in the multi-class setting, resulting in degenerate behavior. Other examples may include instances of non-voting (or tied voting) on patterns, or simply poor inter-class voting

performance by one or more classes that reduces the overall result. In general, strong PGEC results appeared to be very infrequent (i.e., rarely were 'good' teams for all classes (across initializations) seen on the same 'run'), with the likelihood of strong results decreasing with more classes.

An obvious difference with respect to inter-class team behavior between PGEC and CMGE is that CMGE learners (of all classes) evolve together, seeing all of the various point archives at once, whereas PGEC learners can only see one point archive per class-specific initialization. Even though the PGEC point archive is (like CMGE) balanced with respect to in versus out-of-class points, the out-of-class points seen by the various learner populations do not necessarily carry any significance in terms of the inter-class team behaviors. In other words the PGEC learners have, for example, no incentive to learn to 'pass' on points that are being correctly handled by the potential team members from other classes.

Further discussion of team behavior including a comparison of the intra-class behaviors of PGEC and CMGE will be provided in Chapter 8, where evidence for 'weak learner' versus 'problem decomposition' behaviors will be established for PGEC and CMGE, respectively.

5.5 Summary

We began our evaluation of GE methods by clearly establishing that canonical GE, trained as a binary classifier over all training exemplars using and SSE-based fitness function, represents a rather naïve model for classification (Section 5.1). This result has as much to do with the 'sampling' scheme as the classifier model or fitness function, with the accuracy metric being minimized but the much more discerning 'score' metric reflecting the degenerate nature of the ensuing models. Sections 5.2 and 5.3 introduce a constant evaluation limit and repeat the comparison with StdGE (iterate over all fitness cases), RssGE (balanced sub set selection with uniform sampling), and CMGE models. It now becomes very clear that the RssGE represents a significant improvement over the StdGE model on the more informative 'score' metric, where StdGE naturally performs very poorly. However, it is the CMGE models which simultaneously optimize the both the accuracy and score metrics, providing the vast

majority of statistically significant results under either metric. Moreover, the simplicity of the CMGE models (despite being multi-individual models) is typically lower per individual than Std or RssGE. From the computational cost perspective, the additional pass through subsets incurred by the CMGE clustering step in addition to the multiple objective calculation and the management of archives naturally results in longer run times relative to the RssGE and StdGE models but all algorithms are many orders of magnitude faster than canonical GE.

Chapter 6

Results of Neural Network Comparisons

Results presented in this chapter for experiments E6 and E7 (Table 4.6) compare the current CMGE framework with linear (LP) and non-linear (MLP) Artificial Neural Network (ANN) classifiers in terms of overall performance. These results provide the basis for comparisons between the Competitive Multi-Objective GEs (CMGE1 and CMGE2) over 12 classification problems. RssGE ranks are also included in the comparison plots; however, as a baseline indicator only. Results will be discussed in the following sections, using summary tables indicating performance rankings on overall classification in terms of accuracy and score on both train and test data. Bold rankings under the CMGE headings denote statistically significant improvements over the best performing model between MLP or LP according to both F-Test and Kruskal Wallis multiple comparison plots. Similarly, bold rankings under the LP or MLP headings represent statistically significant improvements from the best performing CMGE model. Underlined rankings indicate results that are statistically significant improvements over all other models. The lower portion of each rank summary table provides performance summaries, in terms of sum of ranks (Total), average rank (Avg.), standard deviation of rank (Std.), number of best rankings (N_1), number of worst ranks (N_5), number of bold ranks (**N**) and number of underlined ranks (<u>N</u>). Where appropriate, quartile summaries, box plots and multiple comparison charts will be provided in addition to the rank summary tables in order to highlight representative (or atypical) results in detail on each performance metric. Full ANN quartile summary listings are provided in Appendix G, Section G.4 and the full set of ANN vs. CMGE comparison results for each data set are provided in Appendix B.

6.1 Overall Classification Performance

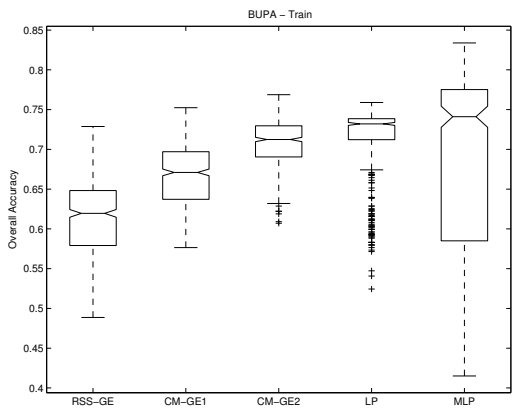
6.1.1 Accuracy

Ranks of the median training accuracies are provided in Table 6.1. LP and MLP Neural Network classifiers receive 20 of the top 24 ranks returned in the training accuracy results, with MLP twice receiving the worst rank (on the IMAG and IRIS problems), and the LP configuration receiving the fourth rank on IRIS. While the Neural Network models typically returned the top two ranking accuracy results on the training data, only half of the time were these results significantly better than both of the CMGE models. Of the CMGE models, CMGE2 appeared to provide the best training context under the overall accuracy metric, twice receiving the top rank and once being significantly better than either ANN approach. Whereas the RssGE baseline was the lowest ranking of all approaches (with an average rank of 4.75), the MLP had the most variation in its ranks at 1.4. Typically the MLP either provided a top ranked training result or the worst (in two cases).

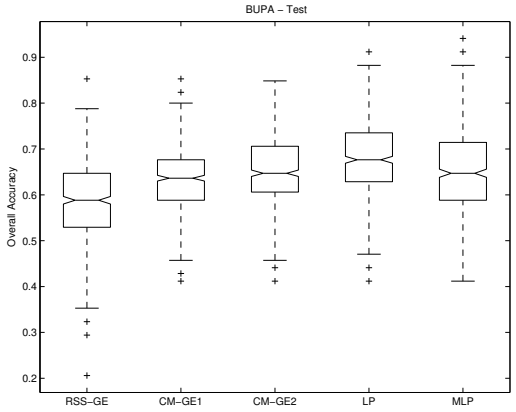
Table 6.1: Comparisons with ANN: accuracy ranks (train)

Data set	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
BOST	5	4	3	**2**	**1**
BUPA	5	4	3	**2**	**1**
CENS	5	4	3	1	2
CONT	5	4	3	**1**	**2**
IMAG	4	2	3	1	5
IRIS	3	**2**	**1**	4	5
KD99	5	4	3	2	1
PIMA	5	3	4	**1**	**2**
SHUT	5	4	1	3	2
THYD	5	4	3	1	2
WINE	5	3	4	**1**	**2**
WISC	5	4	3	2	**1**
Stat	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
Total	57	42	34	21	26
Avg.	4.75	3.5	2.83	1.75	2.17
Std.	0.62	0.8	0.94	0.97	1.4
N_1	0	0	2	6	4
N_5	10	0	0	0	2
N	0	1	1	5	6
$\underline{N}$	0	0	0	3	2

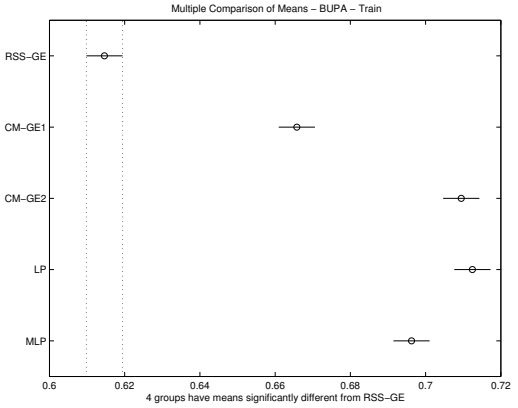
Results in terms of the test accuracy ranks are presented in Table 6.2. Test results were similarly distributed to those observed in the training accuracy results discussed



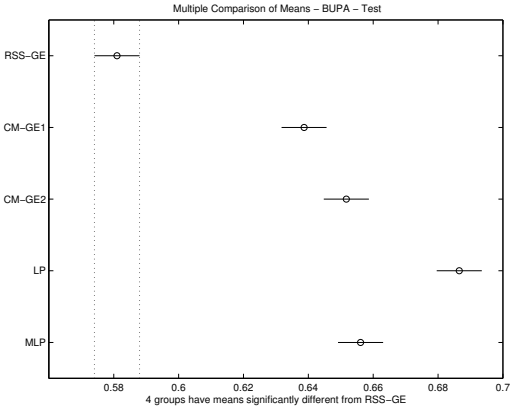
(a) Overall Accuracy (Train)



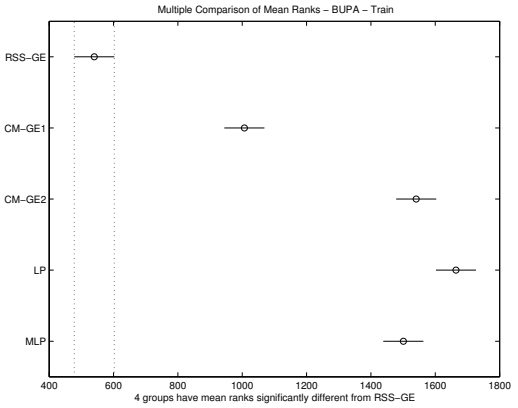
(b) Overall Accuracy (Test)



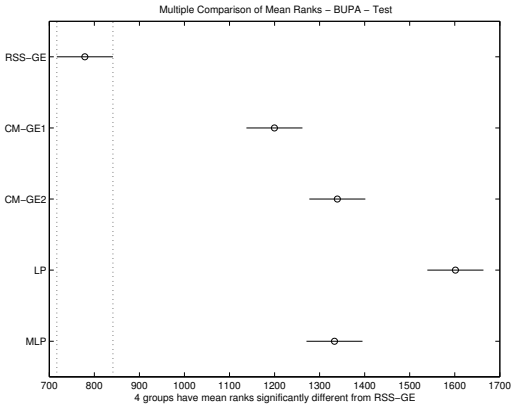
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

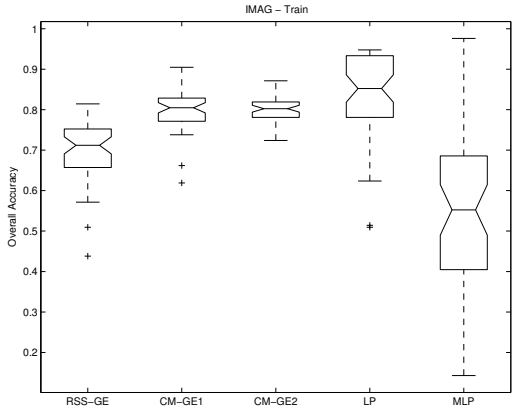


(e) Comparison of Mean Ranks (Train)

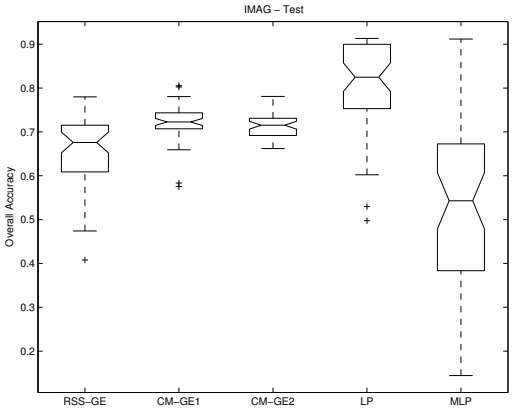


(f) Comparison of Mean Ranks (Test)

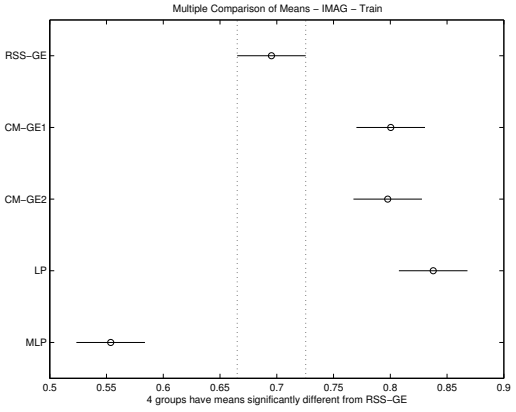
Figure 6.1: ANN comparison of BUPA Overall Accuracy performance.



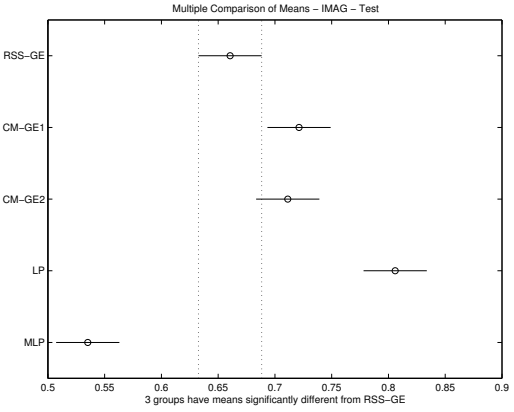
(a) Overall Accuracy (Train)



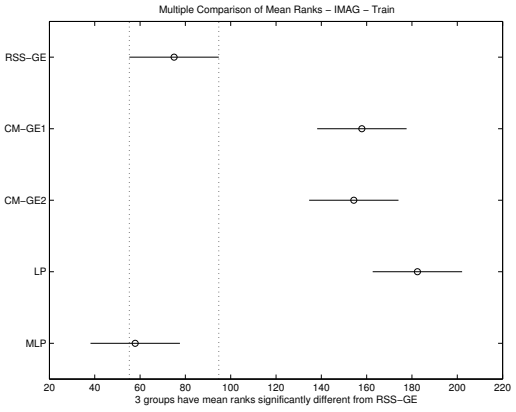
(b) Overall Accuracy (Test)



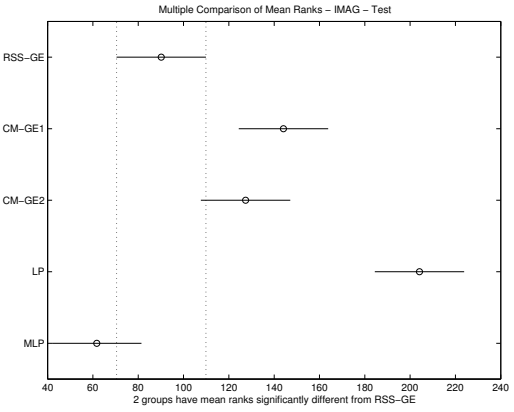
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



(e) Comparison of Mean Ranks (Train)



(f) Comparison of Mean Ranks (Test)

Figure 6.2: ANN comparison of IMAG Overall Accuracy performance.

Table 6.2: Comparisons with ANN: accuracy ranks (test)

Data set	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
BOST	5	3	4	**1**	**2**
BUPA	5	4	2	**1**	3
CENS	5	4	3	1	2
CONT	5	4	3	**1**	**2**
IMAG	4	2	3	**1**	5
IRIS	3	**1**	**2**	4	5
KD99	5	3	4	2	1
PIMA	5	3	4	**1**	**2**
SHUT	5	4	1	3	2
THYD	5	4	3	1	2
WINE	5	3	4	**1**	**2**
WISC	5	4	3	**1**	2
Stat	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
Total	57	39	36	18	30
Avg.	4.75	3.25	3	1.5	2.5
Std.	0.62	0.97	0.95	1	1.24
N_1	0	1	1	9	1
N_5	10	0	0	0	2
N	0	1	1	7	4
$\underline{N}$	0	0	0	7	0

above, with the Neural Network configurations being assigned the top two ranks 19 times out of a possible 24, with 11 of these results being an improvement over CMGE by a statistically significant margin, while 13 of the Neural Network results were not statistically different from the CMGE models. Interestingly, on 7 occasions the basic LP model returned an accuracy that was the outright best among all alternatives, with this being statistically significant at the 95% level based on the multiple-comparison plots. Notably this was never the case for the MLP nor either of the CMGE models although the CMGE approaches were both significant improvements over the Neural Network results on IRIS. Between the MLP and CMGE, only 4 of the 12 data sets provided statistically significant results in favor of the MLP. Similarly to the training results, MLP had the most variation in its ranks (standard deviation of 1.24). Moreover, on 4 of the 12 problems (CENS, KD99, SHUT and THYD) there was not a statistically significant difference between the ANN and CMGE results in terms of test accuracy.

An example that characterizes typical performance is provided in Figure 6.1, where the training results of subplot (a) are not significantly different between the ANN and CMGE models according to the multiple comparison results indicated in subplot (c)

and (e). The BUPA training result in subplot (a) is also typical in that the results show CMGE1 was not able return training accuracies as high as CMGE2; moreover, this represents a significant difference according to the multiple comparison subplots (c) and (e). Despite many outlier points indicating a number of initializations having very low training accuracies, the LP configuration is able to consistently generalize significantly better than any other approach on the BUPA test data, as indicated in subplot (b). Subplots (a) and (b) of Figure 6.1 indicate the typically large amount of variation in the MLP result, particularly in the training results of subplot (a). The results on IMAG (Figure 6.2) again demonstrate the high variability in both ANN results over both train and test (subplots (a) and (b), respectively) with MLP returning the worst results and by far the most spread. While there was no significant difference in the training between LP and the CMGE models, the LP results on test (subplot (b)) again indicate better generalization in terms of the accuracy metric.

6.1.2 Score

Ranks of the class-balanced score metric results are provided in Table 6.3. In contrast to the results based on training accuracy, score ranks were more evenly distributed between the CMGE and ANN classifiers; in general the score may provide a better indication of multi-class performance with this being particularly true in the present case since many of the problems chosen for this work involve a large degree of class imbalance. Of the 24 top ranks, 10 were assigned to CMGE results (5 to each configuration) and 12 were assigned to ANN results (8 to the LP and 4 to MLP), while the RssGE received the final two top rankings on CENS and CONT with training results on CENS being significantly better than any other classifier. The LP results, on the surface, appears to provide the best training context, with an average ranking of 2.25 and on 4 occasions provide the best results of any classifier by a significant margin; however, the variation in this result is the highest, at 1.6, in part because of twice returning the worst result among all classifiers. Moreover, the LP classifier had only 1 result that was significantly better than the CMGE results when it did not have the best result of all classifiers, and it returned results that were significantly worse or statistically no different from CMGE on more than half of all data sets. In terms

of average rank, CMGE1 had the next highest result (2.67) with the second lowest standard deviation (1.23), while having a rank of 1 or 2 on 5 data sets, with three of these results being the top rank and three being statistically significant. CMGE2 provides the next best training result with an average rank of 2.83 and the lowest standard deviation (1.11). Aside from the baseline RssGE approach, the MLP had the worst average rank at 3.42 and the most variation in rank at 1.44. This inconsistency is interestingly demonstrated by the fact that on 4 data sets MLP had the worst result and on 4 other problems it had results that were significantly better than the CMGE approaches. In general, the LP and MLP classifiers either performed very well on the training data or else they returned results that tended to be the worst.

Table 6.3: Comparisons with ANN: score ranks (train)

Data set	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
BOST	5	4	3	**1**	**2**
BUPA	5	4	2	1	3
CENS	**1**	3	4	**2**	5
CONT	2	3	5	**1**	4
IMAG	4	2	3	1	5
IRIS	3	**2**	**1**	4	5
KD99	4	1	2	5	3
PIMA	5	4	3	**1**	**2**
SHUT	3	**1**	**2**	5	4
THYD	4	**1**	**2**	3	5
WINE	5	3	4	**1**	**2**
WISC	5	4	3	2	**1**
Stat	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
Total	46	32	34	27	41
Avg.	3.83	2.67	2.83	2.25	3.42
Std.	1.34	1.23	1.11	1.6	1.44
N_1	1	3	1	6	1
N_5	5	0	1	2	4
N	1	3	3	5	4
$\underline{\mathrm{N}}$	1	0	0	4	0

Test results in terms of the score metric are summarized by the performance ranks in Table 6.4. In large part, the test results on the score metric are similar to those of score on training, with the best generalization being the LP and CMGE models depending on the data set. The one exception to this being the RssGE baseline classifier on CENS which again returned results that were significantly and appreciably better than any other approach. Aside from this result, the baseline was approximately the worst in terms of generalization with half of all data sets resulting

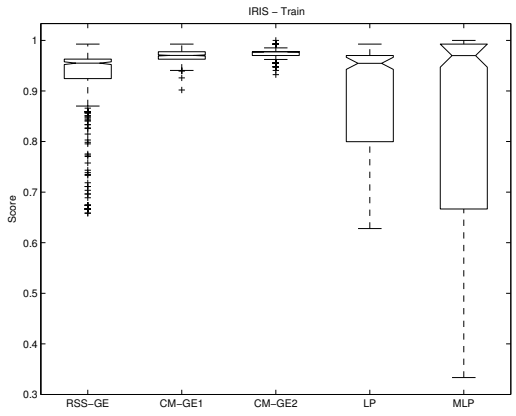
Table 6.4: Comparisons with ANN: score ranks (test)

Data set	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
BOST	5	3	4	**1**	**2**
BUPA	5	4	2	**1**	3
CENS	**1**	3	4	2	5
CONT	2	4	5	**1**	3
IMAG	4	2	3	**1**	5
IRIS	3	**1**	**2**	4	5
KD99	5	1	3	4	2
PIMA	5	4	3	**1**	**2**
SHUT	3	**2**	**1**	5	4
THYD	4	**1**	**2**	3	5
WINE	5	3	4	**1**	**2**
WISC	5	4	3	**1**	2
Stat	**Rss-GE**	**CMGE1**	**CMGE2**	**LP**	**MLP**
Total	47	32	36	25	40
Avg.	3.92	2.67	3	2.08	3.33
Std.	1.38	1.23	1.13	1.51	1.37
N_1	1	3	1	7	0
N_5	6	0	1	1	4
N	1	3	3	7	3
$\underline{\mathrm{N}}$	1	0	0	7	0

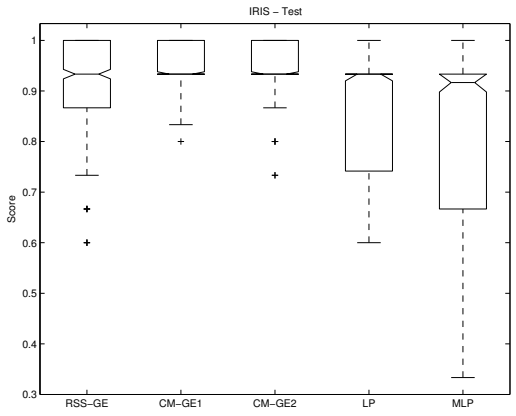
in the lowest rank for this classifier.

In similar results to training, the LP classifier either provided the best generalization context by a difference that was statistically significant from all other classifiers (7 cases) or else it returned results that were approximately the worst, resulting in a low average rank of 2.08 but the largest standard deviation of 1.51. When the LP failed to produce the best overall result, it was never significantly better than the CMGE results. Moreover, LP returned the worst results overall on the highly unbalanced 7-class Shuttle problem, while CMGE2 and CMGE1 provided the best and second best results on this data set, respectively. CMGE1 had the next highest average rank at 2.67 with the second least variation in rank of 1.23, followed closely by CMGE2 with an average of 3 and the lowest standard deviation of 1.13. The CMGE models provided statistically significant improvements over the ANN configurations on IRIS, SHUT and THYD. Aside from the baseline, the MLP results typically provided the worst generalization with an average rank of 3.33 and having a rank of 5 on 4 of the data sets while never returning the top rank.

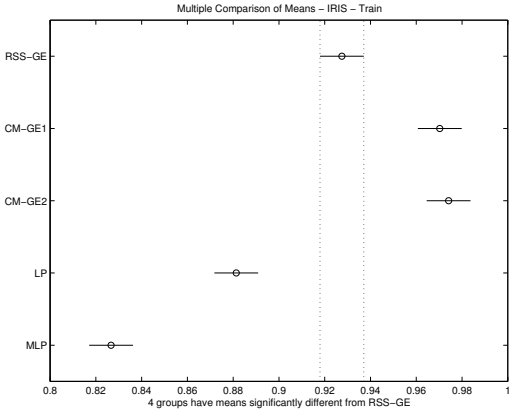
Detailed score comparisons for train and test are provided in Figures 6.3, 6.4,



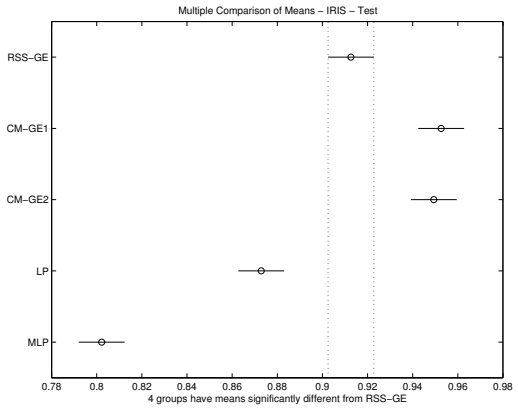
(a) Score (Train)



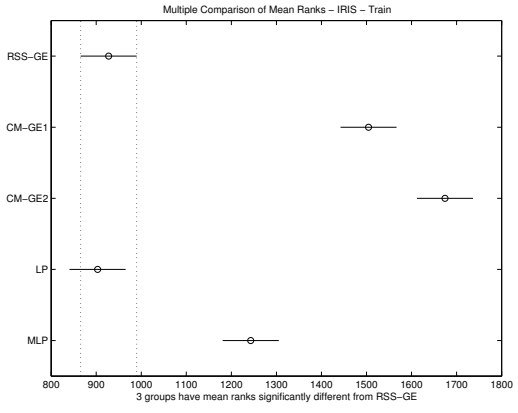
(b) Score (Test)



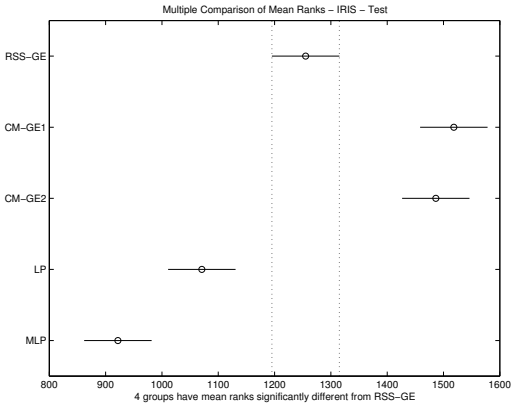
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

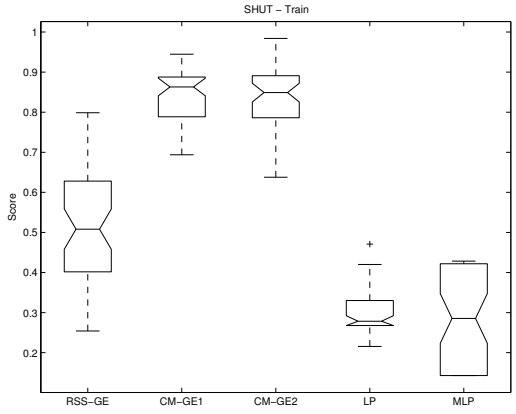


(e) Comparison of Mean Ranks (Train)

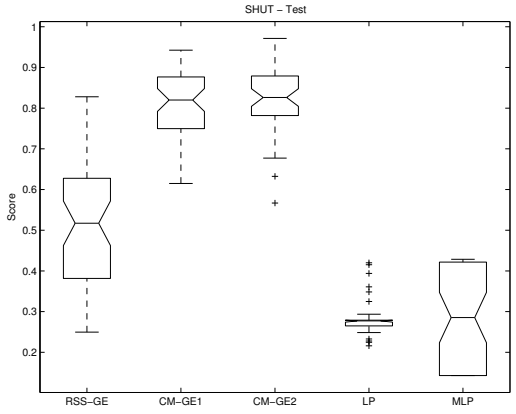


(f) Comparison of Mean Ranks (Test)

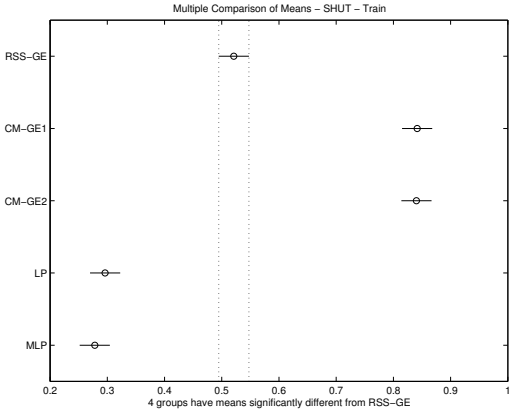
Figure 6.3: ANN comparison of IRIS Score performance.



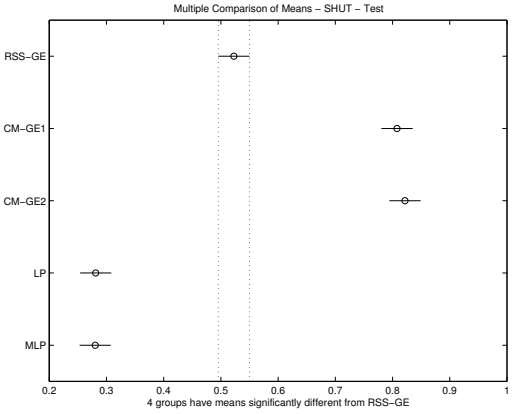
(a) Score (Train)



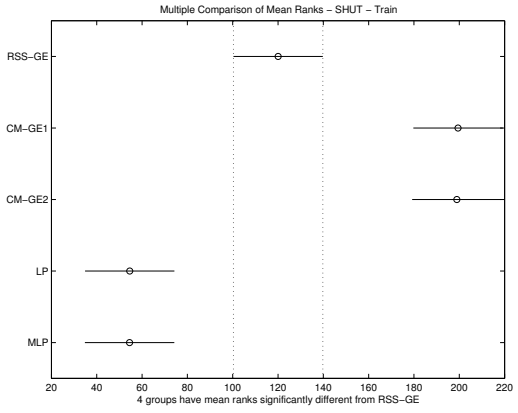
(b) Score (Test)



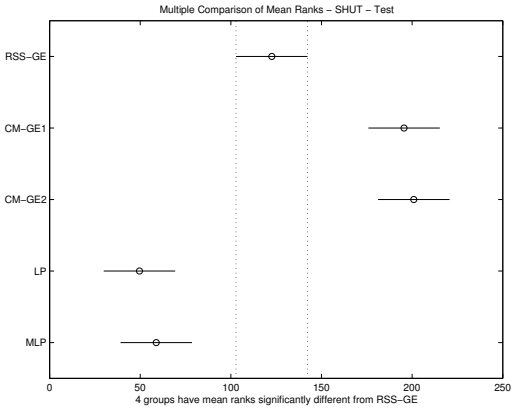
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

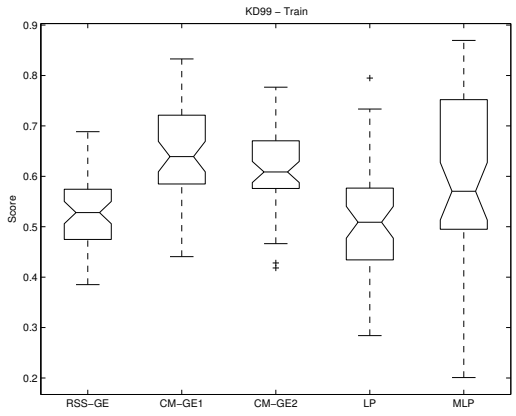


(e) Comparison of Mean Ranks (Train)

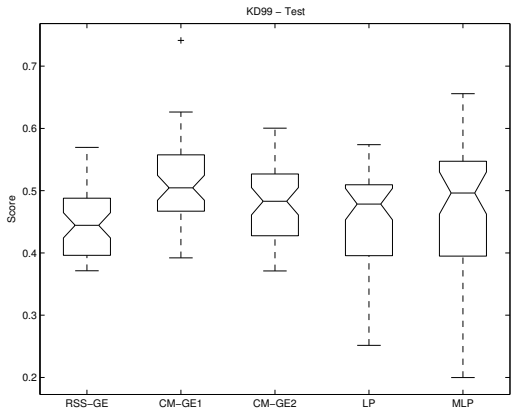


(f) Comparison of Mean Ranks (Test)

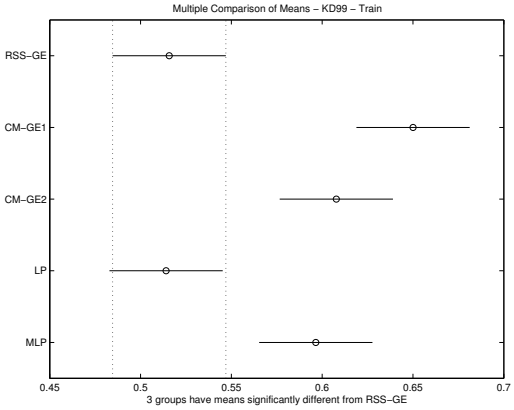
Figure 6.4: ANN comparison of SHUT Score performance.



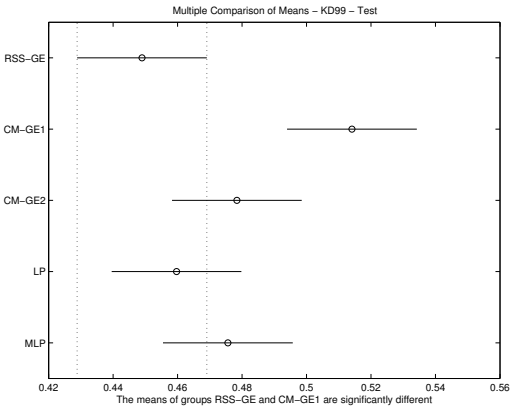
(a) Score (Train)



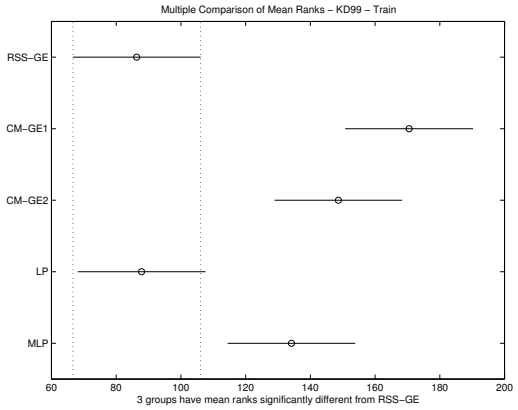
(b) Score (Test)



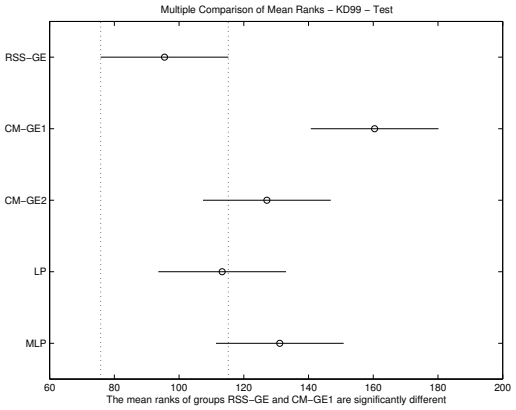
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



(e) Comparison of Mean Ranks (Train)



(f) Comparison of Mean Ranks (Test)

Figure 6.5: ANN comparison of KDD Score performance.

and 6.5. The IRIS results of Figure 6.3 is a somewhat atypical example, being unusual in that the CMGE models are both able to outperform the ANN approaches on both train (subplot (a)) and test (subplot (b)). Moreover, the ANN results on test demonstrate a large degree of skew toward the lower quartile while both of the CMGE configurations regularly return better test scores and, in fact, have ideal test results (score = 1.0) on approximately 25% of the initializations. It is surprising that the ANN classifiers did not perform better on this relatively straightforward data set, however this is perhaps not unreasonable given the variability associated with the ANN approaches. A similar result is illustrated by the Shuttle score comparisons presented in Figure 6.4. On this problem, the training results for CMGE1 and CMGE2 were approximately the same, as seen in subplot (a) and showed very little variation as compared to the MLP result. Interestingly neither of the ANN models were able to return scores on train or test that could compete with even the baseline RssGE, while both CMGE models were significantly better on train and test (as shown by multiple comparisons in subplots (c),(e) and (d),(f) respectively) indicating excellent class-wise generalization which can be verified by the class-wise detection rates presented in Section G.3 of Appendix G.

The KDD box plots with multiple comparisons of Figure 6.5 provide a typical score comparison where no significant differences exists between CMGE1 and CMGE2 against the MLP model on subplots (a) and (b), for train and test scores, respectively. Despite the insignificance between these results, large variations in the MLP scores were again observed, particularly with respect to the results of subplot (a) (train). Both CMGE models return significantly better training results as compared to the LP on the KDD problem, as indicated in multiple comparison subplots (c) and (e); however, CMGE2 does not generalize as well as CMGE1 in this case, and fails to provide statistical significance in its improvement over LP on test, indicated in multiple comparison subplots (d) and (f). The CMGE1 model does, however, significantly outperform the LP classifier on the KDD problem, which fails to outperform the RssGE baseline, indicated by the overlap in multiple comparison groups in subplots (c) and (e) on train, (d) and (f) on test.

6.2 Summary

Benchmarking is performed against two feed forward NN architectures in which the conjugate gradient update rule is utilized. Previous studies [15] [74] limit ANN methods to first order gradient decent methods, despite the drawbacks of such a scheme [69] [10]. In this respect, the ANN models under this study were observed to represent more of a challenge, although the LP model was generally more effective than the non-linear MLP model. Indeed, the LP model either 'worked' or 'failed', thus providing useful feedback in terms of the consistency of a linear model bias; whereas the MLP model was exceptionally inconsistent, typically failing to 'follow up' on the first ranked performance of the LP model. Moreover, cases where the LP might have been expected to perform well (i.e., the 'easier' IRIS data set), the model was not ranked highly; conversely on 'difficult' data sets that might have benefited from a non-linear model, the LP model received the highest rank e.g.,, BOST, BUPA, CONT, PIMA. Conversely, the CMGE models provide relatively consistent performance across all data sets. This is an interesting result, given the inherently stochastic basis for the EC paradigm, and a reflection of the ability of the CMGE multi-objective fitness function to direct credit assignment to where it is most required. The RssGE algorithm maintains the previously observed speciality towards the CENS problem, but results in the largest number of lowest ranked solutions, again emphasizing the contributions apparent in CMGE towards correcting the drawbacks in GE as a classification paradigm.

Chapter 7

Results of Deterministic Comparisons

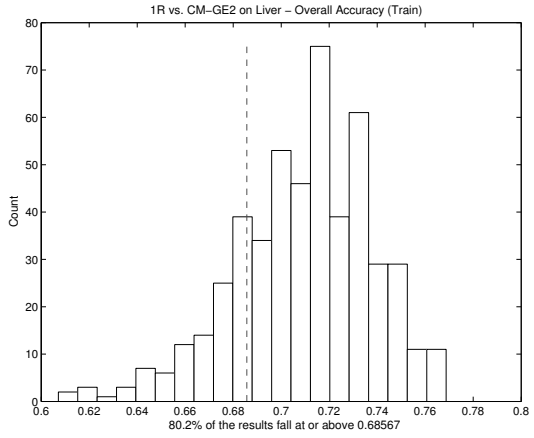
This chapter describes the results of experiments E8 - E10 (Table 4.6), used to compare the proposed CMGE framework with three widely employed and readily available deterministic classifiers (OneR, Naïve Bayes, and C4.5) over the 12-problem classification benchmark. Comparisons between deterministic and non-deterministic (GP) models are not directly accessible since all GP results require multiple independent initializations while deterministic classifiers, by definition, always return the same result given the same collection of exemplars. As a direct consequence, statistical analyses of result distributions, such as the those provided in previous chapters, are not immediately applicable and the following deterministic comparison framework is therefore proposed. All comparisons between deterministic and GP classifiers will be provided through the use of single-result baselines returned by each of the three aforementioned deterministic classifiers. In the case of problems requiring ten-fold cross validation the median deterministic result, calculated over the 10 partition results, establishes the deterministic performance baseline. Comparisons are drawn by plotting the distribution of multiple initializations of the GP classifier for the same metric and recording the total number of results being equal to, or improving on the baseline. This method provides a straightforward means to estimate the probability of any given GP initialization returning a result that equals or improves upon the baseline classifier. A more formal description of this comparison framework is provided in Section 4.5.3, of Chapter 4. The comparative results begin with a graphical comparison of the CMGE2 framework and OneR classifier on the difficult two-class BUPA data set. Figures 7.2 and 7.3 illustrate the deterministic comparison process for class-wise metrics (detection rate and false positive rate, respectively), using histogram depictions of the GP distributions with a vertical dashed line to indicate the baseline value. Results on the overall performance metrics, accuracy and score, are

provided in Figure 7.1. Summary results for all deterministic / GP classifier combinations will be provided for overall accuracy and score results. Summary box plots will be employed to indicate the degree of class-wise performance achieved by the CMGE frameworks in comparison to each deterministic approach, while detailed tables of overall as well as class-wise comparative results are provided in Appendix C. Entries in all tabular summaries of deterministic comparisons indicate the percentage of GP initializations that equal or improve on the deterministic classifier in question. Values between 25% and 50% are highlighted in bold, while results above 50% are underlined. Improvement is defined by larger values in the case of overall accuracy, score and detection rate and smaller values in false positive rate. Quartile summaries of all deterministic classifier performances are provided in Appendix G, Section G.5.

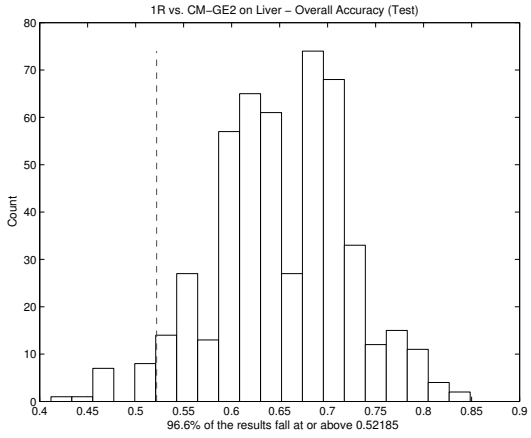
A typical analysis of CMGE's overall performance against OneR is provided in Figure 7.1. Results on overall accuracy in subplots (a) and (b) indicate consistency between train and test results, respectively with CMGE2 matching or outperforming OneR approximately 68% of the time on training data and nearly 97% of the time on test. While the OneR classifier returns a median training accuracy of only 0.69 over the 10 folds, the CMGE2 results are clearly clustered around 0.72, with the upper tail of the distribution reaching 0.76. In terms of test results of subplot (b), the OneR is unable to generalize, effectively returning a 50% accuracy classifier while most of the CMGE2 accuracy results appear to fall above 0.65 and extend beyond 0.8. This analysis provides a good indication that a typical initialization of the CMGE2 classifier will perform much better than OneR in terms of accuracy.

The accuracy results are supported by the score comparison of Figure 7.1, subplots (c) and (d) for training and test scores, respectively. Nearly 70% of the CMGE2 initializations result in improvements on training data while 92% improve on test in terms of score. Moreover the distributions of CMGE2 results in both train and test indicate that the results are typically substantial with the distribution centers appearing well beyond the baselines, particularly with respect to the generalization ability on test.

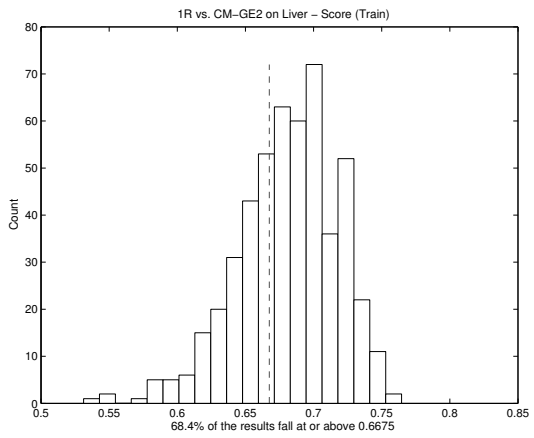
A detailed view of the CMGE2 performance is provided by the class-wise comparisons of Figures 7.2 and 7.3 for detection and false positive rates respectively. Figure



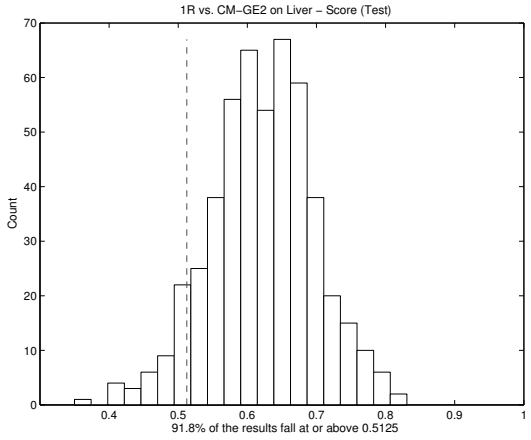
(a) Overall Accuracy (Train)



(b) Overall Accuracy (Test)

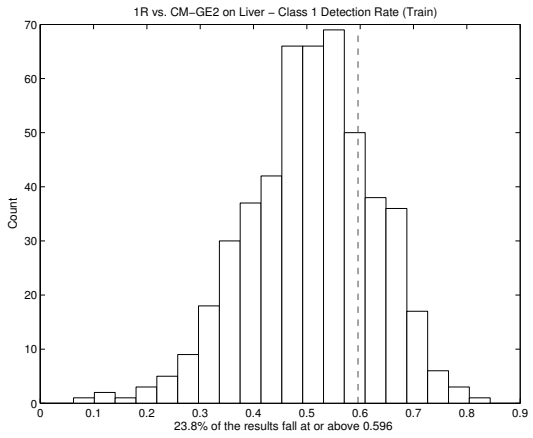


(c) Score (Train)

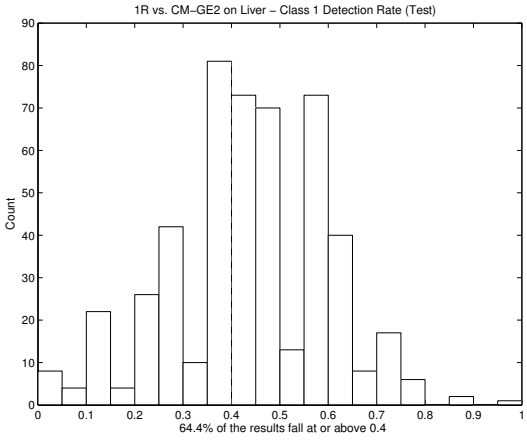


(d) Score (Test)

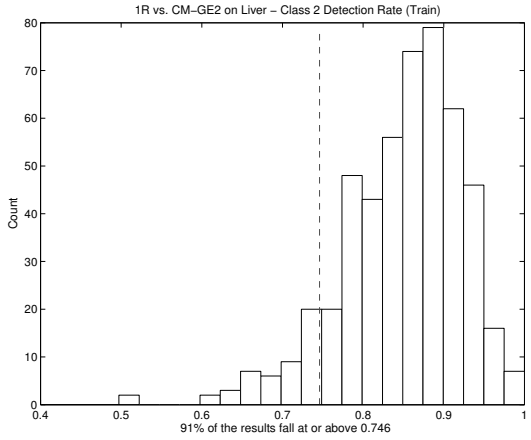
Figure 7.1: Comparison of accuracy and score results. Subplots (a),(b) for Class 1 (Train, Test) and (c),(d) Class 2 (Train, Test). Histograms indicate distribution of GP results and percentage of initializations providing equal or improved values over deterministic classifier OneR on the BUPA data set.



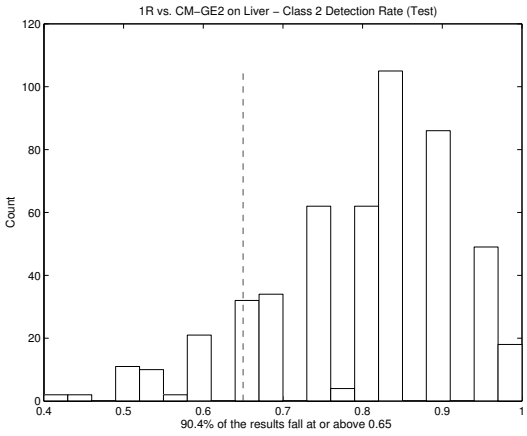
(a) Class 1 Detection Rate (Train)



(b) Class 1 Detection Rate (Test)

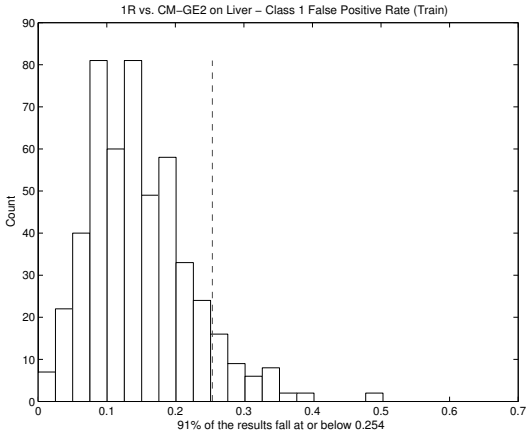


(c) Class 2 Detection Rate (Train)

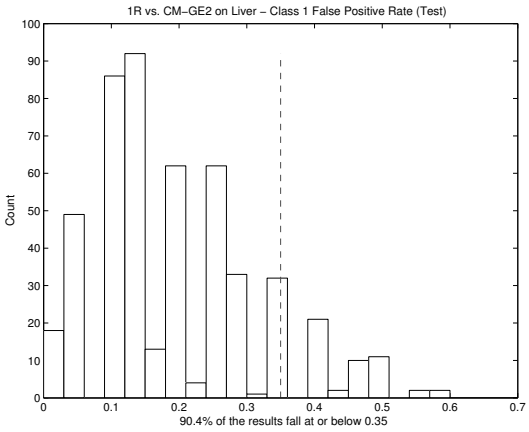


(d) Class 2 Detection Rate (Test)

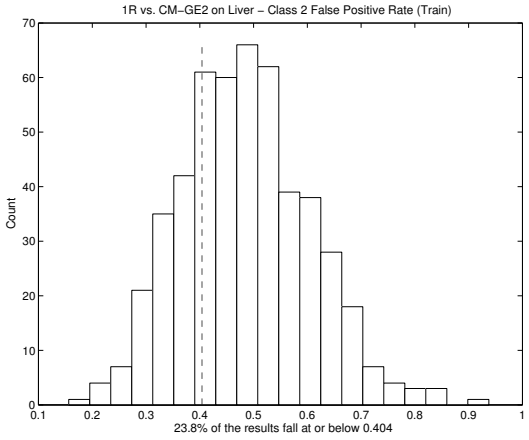
Figure 7.2: Comparison of class-wise detection rates. Subplots (a),(b) for Class 1 (Train, Test) and (c),(d) Class 2 (Train, Test). Histograms indicate distribution of GP results and percentage of initializations providing equal or improved values over deterministic classifier OneR on the BUPA data set.



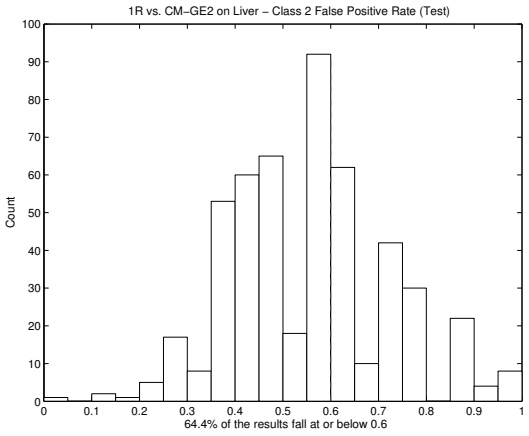
(a) Class 1 False Positive Rate (Train)



(b) Class 1 False Positive Rate (Test)



(c) Class 2 False Positive Rate (Train)



(d) Class 2 False Positive Rate (Test)

Figure 7.3: Comparison of class-wise false positive rates. Subplots (a),(b) for Class 1 (Train, Test) and (c),(d) Class 2 (Train, Test). Histograms indicate distribution of GP results and percentage of initializations providing equal or improved values over deterministic classifier OneR on the BUPA data set.

7.2, subplots (a) and (b) provide results on class 1 over train and test with 23% of all initializations showing improvement in detection rates on the training data. However the OneR algorithm is not able to generalize as well as CMGE2, returning a detection rate of 0.4 with 64% of initializations being an improvement. A more decisive advantage for CMGE2 is shown on class 2, Figure 7.2 subplots (c) and (d), where more than 90% of all initializations match or improve on OneR regardless of train or test data.

Being a two-class problem, the same advantage for CMGE2 on BUPA is seen from the perspective of false-positive rates in Figure 7.3. Because of the inter-class relationship between detection rate and false positive rate on two-class problems (e.g., detection rate of class 1, Figure 7.2, subplot (a), and false positive rate of class 2, Figure 7.3, subplot (c)), their distributions are horizontal mirrors[1] and therefore provide the same advantages over OneR, expressed differently. This behavior can be observed on all two class problems among the results presented in Appendix C.

Due to the large number of comparisons required for the deterministic procedures, all remaining results have been summarized as overall (accuracy and score) performance summary tables and class-wise (detection and false positive rate) notched box plots and will be discussed on a classifier by classifier basis in the following sections.

7.1 OneR Comparisons

Performance summaries of the current framework against the deterministic OneR classifier in terms of the overall accuracy metric are provided in Tables 7.1 and 7.2 for training and test, respectively. Training results for CMGE1 and CMGE2 are similar, with each providing three instances where performance of OneR is equaled or improved in more than one third of all initializations; moreover the CMGE 1 returns percentages above 50 on WISC, IMAG, IRIS, and WINE (all initializations matched or improved on OneR). The same accuracy performance applies to CMGE2, which additionally returns equal or better training results on 80.2% of all initializations on BUPA.

[1]The histograms provided do not depict exactly mirrored distributions because of the binning of the plotting software.

Table 7.1: 1R Comparison - Overall Accuracy (Train)

Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	1	0	0	0.2
WISC	71	61.6	93	98.8
CENS	0	0	0	0
CONT	7	8.2	**44.6**	**45.8**
IMAG	70	68	96	100
IRIS	57.6	**43.2**	90.6	95.2
KD99	0	4	**40**	**46**
BUPA	19.4	5	**36.6**	80.2
PIMA	0.6	0	6.4	5.6
SHUT	0	2	18	**44**
THYD	0	0	2	8
WINE	76.8	66.2	100	100

Table 7.2: 1R Comparison - Overall Accuracy (Test)

Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	8.8	8.6	11	10
WISC	70	67	82.8	90.6
CENS	0	0	0	0
CONT	18.4	18.8	**39.4**	**44.2**
IMAG	84	88	98	100
IRIS	62.4	60.6	**47.2**	**44.8**
KD99	0	0	10	8
BUPA	95.2	76.8	96.8	96.6
PIMA	7.6	10	**45**	**38**
SHUT	0	2	18	**44**
THYD	0	0	4	8
WINE	84.4	79.4	98.8	97

In terms of the overall test accuracy results of Table 7.2, performance remains highly consistent and both CMGE models perform strongly relative to OneR on all but the CENS, KD99 and THYD data sets. Notably, both CMGE models provide some fraction of results that improve on OneR for all data sets other than CENS. Of these, the best results include BUPA, IMAG and WINE where both CMGE models are equal or favorable to OneR in more than 96% of all initializations. CONT, PIMA and IRIS also provide very strong results for both CMGE models (returning percentages of at least 38% for each). CMGE2 additionally provides strong results on SHUT, being equal or improving on OneR on 44% of its initializations.

Table 7.3: 1R Comparison - Score (Train)

Data Set	StdGE	RssGE	CMGE1	CMGE2
BOST	1	0	0	0.2
WISC	79.2	74.2	94.6	99.8
CENS	0	100	**50**	**50**
CONT	24.2	86.8	78.6	70.6
IMAG	70	68	96	100
IRIS	58.4	**43.8**	90.8	95.2
KD99	2	76	98	94
BUPA	10.2	7.2	23	68.4
PIMA	1.2	5.6	8.8	19
SHUT	0	72	100	100
THYD	0	2	**46**	**36**
WINE	77.2	68	100	99.8

Table 7.4: 1R Comparison - Score (Test)

Data Set	StdGE	RssGE	CMGE1	CMGE2
BOST	8	7.6	13.8	12.4
WISC	80.4	80.2	89.8	97
CENS	0	100	52	52
CONT	**30.2**	75.6	63.8	59.4
IMAG	82	88	98	100
IRIS	69.2	67.4	80.6	78.8
KD99	0	**28**	64	**48**
BUPA	72	77.6	79.2	91.8
PIMA	5.4	23.6	**35.2**	**34.8**
SHUT	0	70	100	100
THYD	0	8	58	**42**
WINE	76.8	70	94.8	92.4

Performance summaries of the CMGE models against the deterministic OneR classifier in terms of score are provided in Tables 7.3 and 7.4 for training and test,

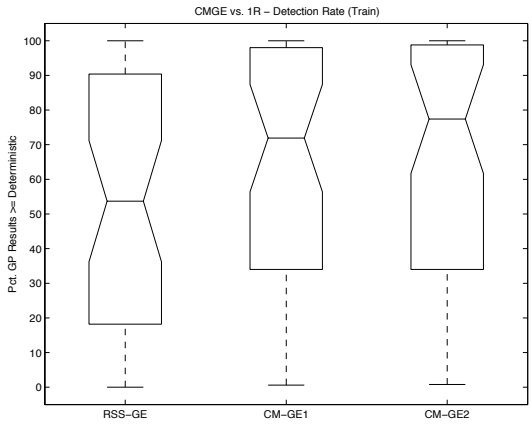
respectively. Both of the CMGE models again provide very strong training contexts when compared with the OneR results. Of the 24 training summary results against OneR, CMGE returned percentages of equal or improved performance of at least 50 on 19 occasions; more than half of these were above 90%. The notable exceptions to the strong training results were BOST and PIMA, where OneR appeared to provide competitive training results. Of the two, CMGE2 appeared to provide more training improvements over the deterministic classifier with both of the difficult PIMA and BUPA problems indicating substantial preference over CMGE1. In both cases, CMGE2 returned equal or improved results over OneR approximately two to three times as often as CMGE1.

Test results in terms of score, provided in Table 7.4, provide additional support of the CMGE model's strong performance against the OneR classifier, with all data sets showing some percentage of initializations resulting in test score improvements for both CMGE models. Of the 24 results, 18 equal or improve on OneR at least 50% of the time with 9 of these being equal or improving at 90% of the time or more. The other 4 results show improvement at least 35% of the time. Only the BOST data set was competitive, with CMGE models returning equal or improved results on at least 10% of initializations. Between the two, CMGE1 appears to have again provided the better generalization context, with only 3 data sets returning comparative results in favor of CMGE2, despite its slight advantage on the training results.

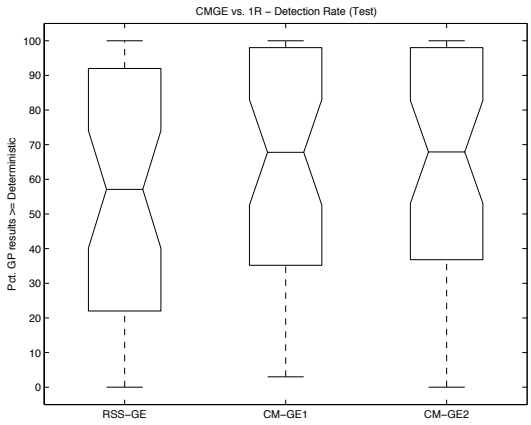
7.1.1 Class-wise Analysis

A class-wise confirmation of the superior results of the CMGE models over the deterministic OneR classifier is presented in Figure 7.4. Notched box plots present the quartile distributions of equal or improved results for class-wise detection in subplots (a) and (b) for training and test, respectively. Similarly the quartiles of training and test false positive rates demonstrating equal or improved results are given in subplots (c) and (d), respectively. Detailed tables of all results are supplied problem by problem in Appendix C.

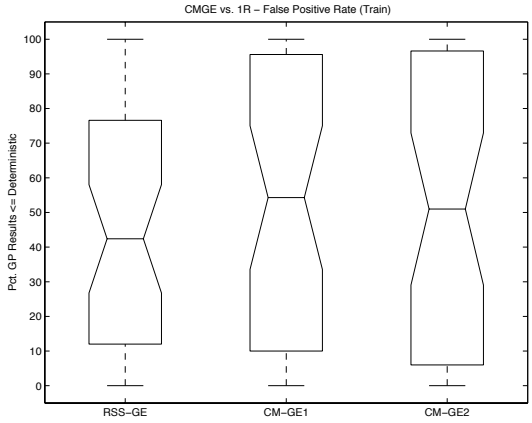
Medians of percentage initializations equal or improving over OneR in terms of



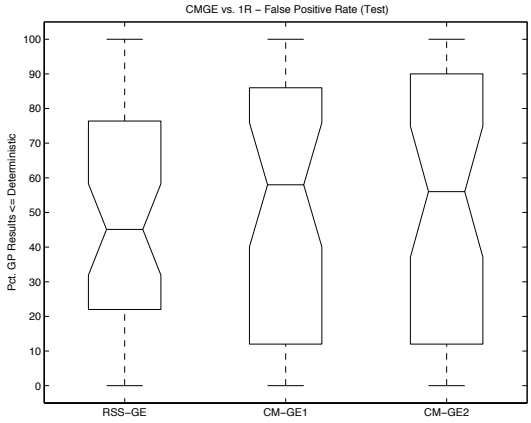
(a) Detection Rates (Train)



(b) Detection Rates (Test)



(c) False Positive Rates (Train)



(d) False Positive Rates (Test)

Figure 7.4: Comparison of class-wise Detection Rate (a), (b) and False Positive Rate (c), (d) as percentage of GP initializations providing equal or improved results over deterministic classifier OneR across data sets D1-D12.

detection rate on training data again indicate an advantage for the CMGE2 algorithm, with a median percentage of approximately 78 as compared to just over 70 for CMGE1. First and third quartiles indicate similar results between the CMGE models at each level with first quartiles of approximately 35% and third quartiles nearing 100%. While both CMGE models show considerable improvements over the baseline approach the difference does not appear to be statistically significant. Test results of subplot (b) reveal nearly identical boxes for CMGE1 and CMGE2, with both providing robust generalization over OneR. The third quartile results indicate that about one quarter of the time the CMGE models are able to return equal or improved results on 95% of their initializations in terms of test detection rates. The median percentage of favorable initializations returned for detection is nearly 70, while the lower quartile is approximately 35.

Class-wise false positive rates appear provide strong support for the superiority of the CMGE models over the deterministic OneR classifier, with upper quartile results indicating approximately 95% of initializations having equal or improved performance on the training data. Medians in both CMGE frameworks are above 50%, however the CMGE1 appears to hold a slight advantage over CMGE2 in this case. The lower quartile results are similar between the CMGE models, at approximately 10% which is not a particularly strong result as it indicates that some initializations having improved results in detection may actually be over focusing and causing these false positives.

Test results in terms of false positive rates indicated in subplot (d) are again strong positive indications of the CMGE performance in comparison to OneR. The CMGE1 approach appears to provide the slightly superior false positive rate test results in comparison to the deterministic result of OneR; however, both CMGE medians indicate equal or improved results being returned by more than 50% of initializations each, with upper quartile results being approximately 85% of initializations for CMGE1 and 90% for CMGE2. While all CMGE results appear to provide an appreciable improvement over the baseline, the statistical significance of the difference is unclear based on the notches provided in the box plots.

7.2 Naïve Bayes Comparisons

Performance summaries of the current framework against the deterministic NB classifier in terms of the overall accuracy metric are provided in Tables 7.5 and 7.6 for training and test, respectively. Recall from Section 4.5.3 that the tables report the percentage of CMGE initializations equalling or improving on the deterministic solution. Notably, the CMGE models provide equal or improved results over all 12 data sets on training, though the WINE and PIMA results are tenuous at less than 1% and 5% of all initializations being equal or improved, respectively. The strongest comparative results returned by CMGE1 were on BOST, CENS, IRIS, KD99, BUPA, and SHUT; all corresponded to equal or improved training accuracy on at least one third of the initializations. CMGE2 provided even stronger results on each of these multi-class problems (along with the 2-class BUPA) with a minimum of 45% of initializations while additionally comparing favorably on the WISC problem. In general, and aside from the CENS data set, none of the baseline GPs compared as favorably in comparison to NB. Similar results are observed in terms of test accuracy, however the generalization on the IMAG data set is reduced to 4% in the case of CMGE1 while failing to equal NB entirely in the case of CMGE2. Conversely, the generalization results improved over training in the cases of WINE, PIMA and CONT with a preference for CMGE1 over CMGE2.

Table 7.5: NB Comparison - Overall Accuracy (Train)

Data Set	StdGE	RssGE	CMGE1	CMGE2
BOST	11	7	**37.8**	**44.6**
WISC	**27.8**	22.6	11.2	**37.2**
CENS	100	10	94	94
CONT	3.2	2.8	24.8	21
IMAG	0	0	24	12
IRIS	56.4	**41**	88.2	94
KD99	58	72	94	94
BUPA	99.8	86.4	100	100
PIMA	0.2	0	3	2.8
SHUT	0	4	**32**	56
THYD	0	0	20	22
WINE	0.2	0	0.2	0.2

Score results support the favorable performance of CMGE observed on the overall accuracy metric. CMGE2 typically returned the best training scores, with results

Table 7.6: NB Comparison - Overall Accuracy (Test)

Data Set	StdGE	RssGE	CMGE1	CMGE2
BOST	16	14.8	23.8	**25.4**
WISC	25	20.2	22.6	**28.4**
CENS	100	10	94	94
CONT	12.8	13	**29.6**	**32**
IMAG	0	0	4	0
IRIS	**36.8**	**31**	**47.2**	**44.8**
KD99	20	**28**	66	50
BUPA	88.8	65	92.8	92.8
PIMA	3.2	5.2	24.6	16.6
SHUT	0	4	**32**	52
THYD	0	0	16	10
WINE	8.8	6.6	20.2	15.2

on BOST, WISC, and SHUT being equal or better in terms of score at least 25% of the time; moreover, the IRIS, BUPA and THYD comparisons were favorable for CMGE2 on at least 94% of the initializations. Training results on CENS, CONT, KD99 and WINE were mainly preferable for NB, with CMGE2 returning very few classifiers (in fact, none on KD99 and CENS) that matched the training scores of NB. While CMGE1 performance on the training data was typically less dominant than CMGE2, the performances on test scores were largely homologous, with 5 data sets (BOST, WISC, BUPA, SHUT and THYD) indicating a preference for CMGE1 and 6 (CONT, IMAG, IRIS, KD99, PIMA and WINE) providing results in favor or CMGE2. Aside from CENS, CONT, IMAG and KD99, the score test results were strong with 3 data sets showing preferable performance for CMGE models on 25 - 50% of the initializations and 3 indicating preferable initializations at least 50% of the time.

In contrast to the results provided by overall accuracy, the score test results on CENS failed to match NB, which indicates that the NB classifiers were typically able to achieve an improved degree of performance on the minority class of the highly unbalance CENS data set. This is confirmed by examining the NB class-wise detection rates provided in Appendix G, Section G.5, where results are provided as 0.88 for class 1 and 0.63 for class 2, whereas CMGE1 and CMGE2 class-wise detection rates of Appendix G, Section G.5 are provided as 0.98, 0.198 for CMGE1 (class 1, class

Table 7.7: NB Comparison - Score (Train)

Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	10.4	6.8	**39**	**45.2**
WISC	20.8	21.2	6.4	**25.2**
CENS	0	**28**	0	0
CONT	0	2	1.2	0.4
IMAG	0	0	24	12
IRIS	56.4	**41**	88.2	94
KD99	0	0	8	0
BUPA	55.4	69.8	67	97.8
PIMA	0	0.8	0.8	4
SHUT	0	0	58	60
THYD	0	22	94	94
WINE	0	0	0	0.2

Table 7.8: NB Comparison - Score (Test)

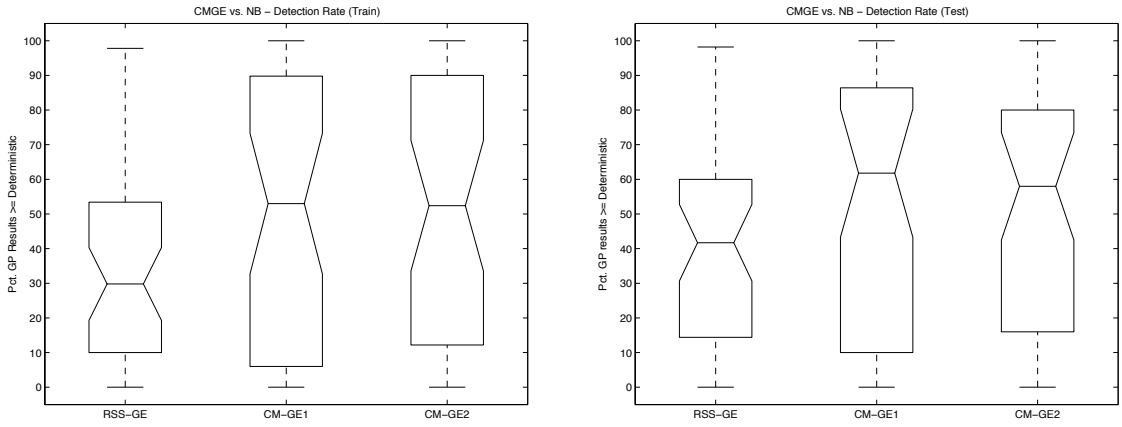
Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	15.2	15	24	**26.4**
WISC	21	19	21.6	**29.6**
CENS	0	20	0	0
CONT	1.6	12.8	7	2
IMAG	0	0	4	0
IRIS	**36.8**	**31**	**47.2**	**44.8**
KD99	0	0	2	0
BUPA	52.6	60	60.8	78.4
PIMA	1.8	9.2	16.6	12.6
SHUT	0	6	60	70
THYD	0	22	90	92
WINE	8.8	6.6	20.2	15.2

2) and 0.986, 0.175 for CMGE2 (class 1, class 2). This result is approximately consistent over train and test data and therefore does not indicate brittle performance on on the minority class; rather it illustrates an over-focusing behavior on class 1 in both CMGE approaches. Given the strong performance of StdGE under the accuracy metric and RssGE under the score metric for the CENS data set, it is clear that the class-distribution provides a difficult test for any of the sub sampling algorithms. This pattern of performance follows the behavior recognized by Weiss and Provost with respect to a C4.5-based sub sampling algorithm [116]. Conversely, CMGE and NB provide solutions that lie between the two extremes established by StdGE and RssGE (accuracy optimization versus score optimization). Notably the NB performance is not dominant in a class-wise comparison, however the score metric identifies a preference for NB that would be relevant for many practical applications. This issue will be revisited in the following class-wise analysis.

7.2.1 Class-wise Analysis

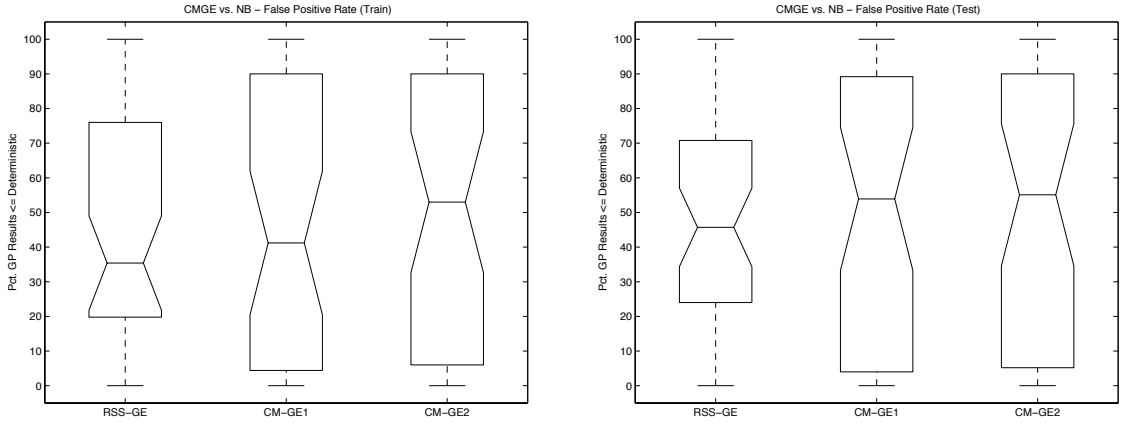
A class-wise confirmation of the predominantly superior results of the CMGE models over the deterministic NB classifier is presented in Figure 7.5. Notched box plots present the quartile distributions of equal or improved results for class-wise detection in subplots (a) and (b) for training and test, respectively. Similarly the quartiles of training and test false positive rates demonstrating equal or improved results are given in subplots (c) and (d), respectively. Detailed tables of all results are supplied problem by problem in Appendix C.

The medians of percentage improvement in detection rate over NB for CMGE1 and CMGE2 in training are very similar, with both values being above 50%; the third quartiles are similarly comparable at approximately 90%. A small advantage in training appears to be demonstrated by the CMGE2 model results in that the lower quartile is clearly above 10%, while CMGE1 is closer to 5%. While the presence of overlapping of notches indicates a non-statistically significant difference, both of the CMGE approaches are clearly more successful than the baseline (RssGE) in matching or improving the NB results in terms of detection in training. Regarding the test results presented in Figure 7.5 subplot (b), the median percentages are indicative



(a) Detection Rates (Train)

(b) Detection Rates (Test)



(c) False Positive Rates (Train)

(d) False Positive Rates (Test)

Figure 7.5: Comparison of class-wise Detection Rate (a), (b) and False Positive Rate (c), (d) as percentage of GP initializations providing equal or improved results over deterministic classifier NB across data sets D1-D12.

of the larger trend; specifically the CMGE1 model provides better generalization in test detection with a median percentage above 60, while the training performance of CMGE2 did not generalize to test as well, returning a median percentage of initializations matching or improving NB below 60. Similarly the third quartile results demonstrate a slightly better degree of generalization in CMGE1 (approximately 85%) as compared to CMGE2 (approximately 80%); this difference is perhaps offset by the modestly improved performance by CMGE2 in the first quartile, however the clear preference is for CMGE1 in terms of generalization of detection rate. Again both CMGE approaches are clearly improvements over the baseline scalable GP approach.

Train and test false positive rates of Figure 7.5, subplots (a) and (b) respectively are similar to those provided by detection rate. In terms of training, CMGE2 medians indicate that half the detection rate comparisons report 55% of initializations that result in equal or favorable performance over NB. This represents a difference of approximately 10% over CMGE1, while the percentages indicated at the first and third quartiles are nearly identical. Test results of subplot (b) show that CMGE1 again gains in terms of performance on CMGE2 as the median values for percentage improvement over NB are approximately equal at 55%. Clearly both CMGE models provide highly favorable class-wise comparisons against the deterministic NB classifier.

7.3 J48 Comparisons

Performance summaries of the current framework against the deterministic J48 (the Weka implementation of the standard C4.5 (revision 8) decision tree) classifier in terms of the overall accuracy metric are provided in Tables 7.9 and 7.10 for training and test, respectively. Based on training result summaries of Table 7.9, it is readily evident that neither the CMGE nor baseline models are able to compete with J48 in terms of training accuracy on the majority of data sets. The two exceptions being the relatively straightforward IRIS and WINE problems, where CMGE1 is able to match or improve on 14.4% of IRIS initializations and CMGE2 has only a slightly better result of 25%, while the WINE results are tenuous at 0.2%.

Despite the obvious disparity in training, the summary results of test provided

Table 7.9: J48 Comparison - Overall Accuracy (Train)

Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	0	0	0	0
WISC	0	0	0	0
CENS	0	0	0	0
CONT	0	0	0	0
IMAG	0	0	0	0
IRIS	9.8	3.8	14.4	25
KD99	0	0	0	0
BUPA	0	0	0	0
PIMA	0	0	0	0
SHUT	0	0	0	0
THYD	0	0	0	0
WINE	0.2	0	0.2	0.2

Table 7.10: J48 Comparison - Overall Accuracy (Test)

Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	7.4	7.2	11	10
WISC	**26.4**	22.2	**32.8**	**40.4**
CENS	0	0	0	0
CONT	9.6	9.8	24.4	20.8
IMAG	0	0	0	0
IRIS	62.4	60.6	**47.2**	**44.8**
KD99	0	0	0	0
BUPA	18.6	12.4	23	**31.8**
PIMA	3	4.6	24.4	16
SHUT	0	0	0	0
THYD	0	0	0	0
WINE	**25.4**	21.8	23.6	18

in Table 7.10 clearly demonstrate that competitive classifiers are returned for more than half of all problems when considering generalization. Specifically, the CMGE1 models return competitive or improved test accuracies more than 20% of the time on WISC, CONT, IRIS, BUPA, PIMA and WINE with WISC and IRIS having at least one third of results in the competitive or improved range. Results are very similar for CMGE2; however, stronger results are provided on WISC, and BUPA. Moreover, both CMGE models are able to return a fraction of test results matching or improving on BOST, however only in approximately 10% of all initializations each.

Table 7.11: J48 Comparison - Score (Train)

Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	0	0	0	0
WISC	0.2	0	0	0
CENS	0	0	0	0
CONT	0	0	0	0
IMAG	0	0	0	0
IRIS	8.8	2.6	12.6	21.8
KD99	0	0	0	0
BUPA	0	0	0	0
PIMA	0	0	0	0
SHUT	0	0	0	8
THYD	0	0	0	0
WINE	0	0	0	0.2

Table 7.12: J48 Comparison - Score (Test)

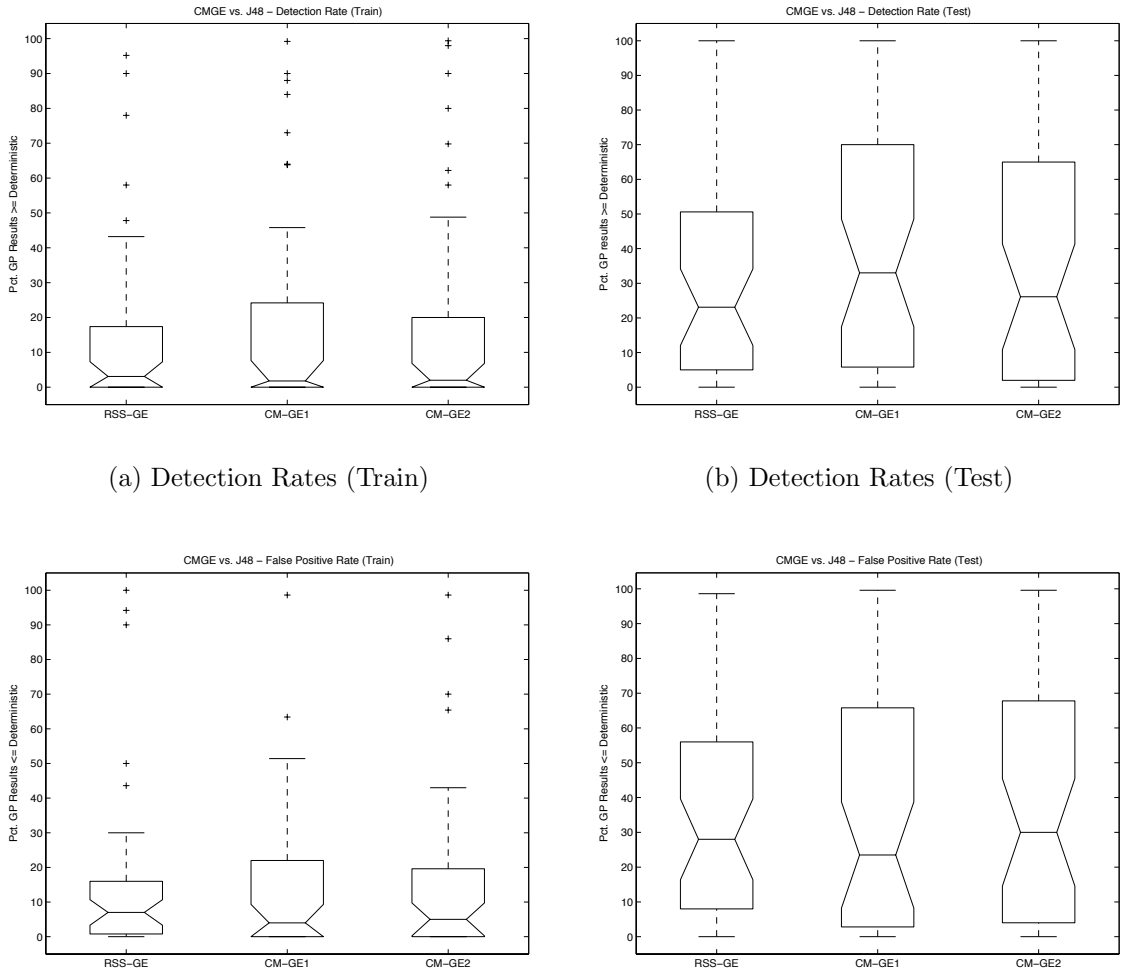
Data Set	**StdGE**	**RssGE**	**CMGE1**	**CMGE2**
BOST	7.2	6.6	12.6	11.4
WISC	**31.8**	**29.8**	**37.2**	**46.6**
CENS	0	74	2	4
CONT	6	23.2	16	9
IMAG	0	0	0	0
IRIS	69.2	67.4	80.6	78.8
KD99	0	0	8	0
BUPA	15.4	14.8	15	24.8
PIMA	1.8	9	16.6	11.4
SHUT	0	0	**36**	**40**
THYD	0	0	0	0
WINE	**29.6**	24.2	58.2	**46.4**

Score results for train and test are provided in Tables 7.11 and 7.12 for training and test, respectively. Once again the training results provided little comparison between the CMGE models and the deterministic J48 classifier. Notably, the CMGE2 is able

to provide some competitive results on the SHUT data set in terms of training scores, with all other results being similar to those discussed in training accuracy, above. The results of test scores of Table 7.12 are more indicative of the practical multi-class performance since the score provides equal weighting to performance on each class so that the class-wise distributions of the data do not affect the contributions to performance on the score statistic. Despite the lower training results, CMGE models are able to compete convincingly in terms of the test scores with results on only two data sets (IMAG and THYD) failing to return competitive or improved results. The strongest comparative results for CMGE1 are WISC, IRIS, SHUT and WINE, all returning equal or improved results with at least 36% of the initializations. Similar generalization performance is returned by CMGE2 with BUPA also returning a result of approximately 25% and again stronger results for WISC (46.6%) and SHUT (40%). While many of the overall results appear to be in favor of the J48 decision tree classifier, the obvious difficulty associated with its performance appears to be related to over learning, where many training exemplars are reliably learned. However, they are learned largely to the extent of J48's detriment as the generalization is clearly reduced to the point that both CMGE models (as well as some of the baseline GP approaches) are able to provide consistently competitive or stronger results.

7.3.1 Class-wise Analysis

A similar trend to that observed in the score results is apparent in the class-wise detection and false positive comparison plots provided in Figure 7.6. While both the detection and false positive statistics (subplots (a) and (c), respectively) indicate typically no better than 25% of initializations improving on either of the CMGE models, test results provided in subplots (b) and (d) indicate that many cases improve on J48's generalization. Of the two CMGE approaches, CMGE1 demonstrates a higher percentage of equal and improved detection rates in test with third quartile of results at 70% as compared to approximately 65% for CMGE2. Similarly, the median and lower quartiles (approximately 35% and 8% respectively) are preferable to those of CMGE2 (near 25% and 5%, respectively). False positive results appear to be nearly the same for both CMGE models with a slight preference for CMGE2



(a) Detection Rates (Train)

(b) Detection Rates (Test)

(c) False Positive Rates (Train)

(d) False Positive Rates (Test)

Figure 7.6: Comparison of class-wise Detection Rate (a), (b) and False Positive Rate (c), (d) as percentage of GP initializations providing equal or improved results over deterministic classifier C4.5 across data sets D1-D12.

in terms of median. These results provide additional evidence of over learning in the J48 classifier and demonstrate the competitive generalization abilities of the CMGE framework.

7.4 Summary

Three widely used deterministic (greedy) machine learning algorithms of increasing complexity have been compared against the baseline Std and RssGE models and CMGE variants. To do so we adopt a unique approach to performance evaluation that directly addresses the question of what the likelihood is of a non-deterministic machine learning model (in this case GE) matching or improving the performance threshold set by the deterministic model. With respect to OneR, the CMGE models perform consistently better, irrespective of the data set; whereas the NB model was able to consistently perform better in the case of three of the 12 classification problems (CENS, IMAG, KD99). Under the strongest deterministic learner, C4.5, CMGE algorithms appeared to provide a distinct advantage under test data. Interestingly, CMGE was more effective under a mixture of the easier (IRIS, WINE, WISC) and more difficult (BUPA, SHUT, PIMA, BOST) data sets. Naturally we do not expect to do better than all algorithms under all data sets all of the time, however, it was naturally gratifying that the EC models were only 'shut out' of two data sets entirely (CONT and THYD). Moreover, CMGE was by far the best EC method relative to the deterministic algorithms considered.

With respect to the baseline EC models of StdGE and RssGE it is interesting that the StdGE algorithm is actually 'shut out' by the OneR heuristic on four of the data sets under both accuracy and score (CENS, KD99, SHUT, THYD). RssGE receives a similar fate under the accuracy metric, although this is understandable when considering the OneR heuristic's implicit bias towards classifying the major class correctly. In short, although the OneR algorithm has obvious deficiencies it is still able to act as a surprisingly effective classifier, as recognized in the original presentation of the algorithm [52]. The performance of NB is much stronger under the score metric than OneR, with StdGE being 'shut out' on IMAG in addition to CENS, KD99, SHUT, and THYD (the latter four also being being the case for OneR)

and other than the CENS data set, RssGE receives a similar fate. C4.5 demonstrates the same tendency, thus all the deterministic algorithms share a common data set (learning) bias. Conversely, the most consistent ANN method from Section 6, the Linear Perceptron, was weakest on IRIS, KD99, SHUT, and THYD thus representing an entirely different learning bias than the deterministic models. The only data set that deterministic and ANN models consistently performed poorly on was IRIS, whereas all EC models appeared to favor this problem.

Chapter 8

Problem Decomposition and Feature Selection

Multi-class classification and automatic intra-class problem decomposition are principal contributions of the present classification framework as laid out in the stated objectives at the introduction of this thesis. The inter-class voting behavior leading to multi-class classification has been demonstrated empirically through the strong team-based results of the CMGE models, presented in Chapter 5. The intra-class coverage behavior leading to problem decomposition will be established in the current chapter and attributed the following aspects of the algorithm design:

- A local (Gaussian) wrapper function;
- Enforcement of class consistency in GP evaluations;
- Explicit cooperation objective;
- Real-valued outcome vectors associated with classification strength;
- Class-wise Parteto-coevolutionary archiving.

These characteristics will be discussed in the context of voting behavior and compared to the binary PGEC classification algorithm in Section 8.1. Moreover, a natural consequence of the CMGE framework is the automatic and problem specific selection of relevant features for classification. This property has significance for both performance in terms of training overhead as well as online (post-training) performance and solution complexity. Section 8.3 will examine this aspect of the current framework in detail and provide examples for selected data sets.

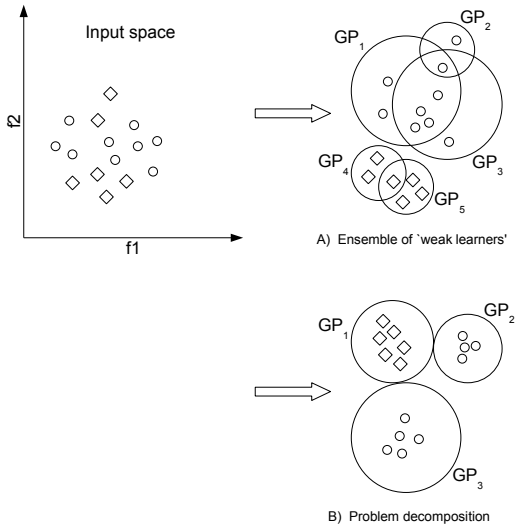


Figure 8.1: Coverage associated with 'weak learning' as opposed to explicitly cooperative, 'orthogonal' coverage corresponding to problem decomposition.

8.1 Intra-class Voting Behavior

Teamwork implies the presence of constructive interactions between individuals, or team members. Informally this refers to non-conflicting inter-class voting and orthogonal intra-class coverage. Indecisive voting between classes is likely to cause collisions or incorrect classifications (particularly under a WTA policy), therefore voting that is accurate in terms of class-consistentcy while supporting a high degree of certainty is expected to be preferable when identifying team members. To this end, real-valued outcomes have been established such that individual / exemplar interactions provide precisely this information in an outcome-vector context to the archiving process. This behavior is driven by a winner-take-all (WTA) voting policy which implies a meaningful voting range over which a 'winning' expression may be defined for a given exemplar. The local (Gaussian) output wrapper provides for this functionality in two ways. First, it defines a real-valued range ([0,1]) in which an individual's output may fall. Second, the enforcement of cluster consistency on individuals (in terms of the Gaussian error evaluation) establishes this output as a meaningful degree of certainty in each classification, where a value near 1 indicates a greater degree of confidence.

The explicit intra-class cooperation objective maintains evolutionary pressure toward individuals that provide coverage in the problem at any stage of training. Specifically, the overlap objective defined against the coverage of the learner archive is designed to guide the search toward individuals that perform well on intra-class subsets of the problem where coverage is weak or non-existent and thus affords a focusing mechanism that promotes the establishment of orthogonal decompositions defining cooperative problem decomposition as opposed to ensembles of 'weak learners' where coverage is a result of multiple (supporting) membership values (or votes) leading to classifications (Figure 8.1).

8.2 Coverage

CMGE's problem decomposition is achieved by application of the WTA voting policy, with the maintenance of relevant team members and the responsibility of remembering progress being guaranteed by the class-wise archiving of the modified IPCA algorithm. The following coverage plots demonstrate CMGE intra-class voting behaviors as histograms of differences between winning individual outputs and the median output of the remaining same-class individuals. These differences in output characterize the pattern coverage among same-class team members where a small difference (near zero) between winner and team output indicates a large degree of overlap in coverage, whereas a large difference (near 1) is associated with a non-overlapping classification. The coverage of CMGE1 and CMGE2 over 9 of the 12 benchmark data sets[1] will be compared with the coverage provided by PGEC, the multi-class GE implementation of Lemczyk's PGPC [71].

A detailed discussion of the significance of the difference cases is provided in the following sections. A selection of representative coverage histograms will be provided in the ensuing comparative analysis, with all remaining coverage histogram results being provided in Appendix D.

8.2.1 Small Median Difference

Small differences (~ 0) can arise in one of two ways:

[1]Due to their large sizes, CENS, KD99 and SHUT have been omitted from the coverage analysis.

1. A strong winning output (~ 1) that has similarly strong median in-class team support;

2. A moderate to low winning output where the choice of winner among in-class team members is small.

While the first of these scenarios indicates support for the winning classification, the second corresponds to a more tenuous classification as the winning output approaches zero.

8.2.2 Moderate Median Difference

A difference near 0.5 can occur in one of two ways:

1. A strong winning output that has comparatively moderate median in-class team support;

2. A moderate winning output with weak or low support from in-class team members.

Intuitively, this mid-range situation involving winner / team differences near 0.5 seems to be the least desirable in the problem decomposition context in that such behavior corresponds to ambiguous or weak decomposition. Moreover, a weak decomposition can lead to a difficult analysis of the solution invoked by the classifier since either partial or weak relationships exist between exemplars and responses (see also Figure 8.1).

8.2.3 Large Median Difference

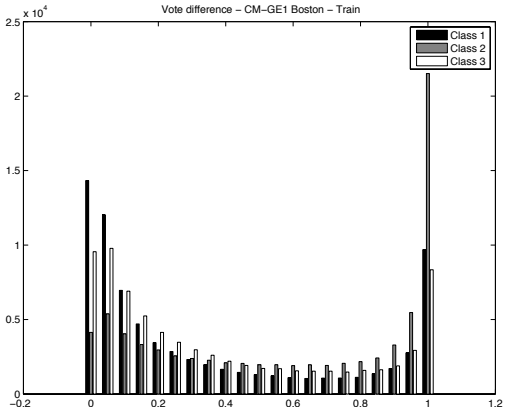
In contrast to the small and moderate difference scenarios, large margins of output difference require a strong output from the winner and low output in median team response, suggesting specialized behavior in the winning individual and indicating strong problem decomposition (disjoint exemplar-wise coverage); moreover, this promotes solution understandability as a single solution can now be associated with a each exemplar. While the current framework specifically aims to encourage this latter behavior, there are no steps taken to explicitly preclude the others.

8.2.4 Comparative Results

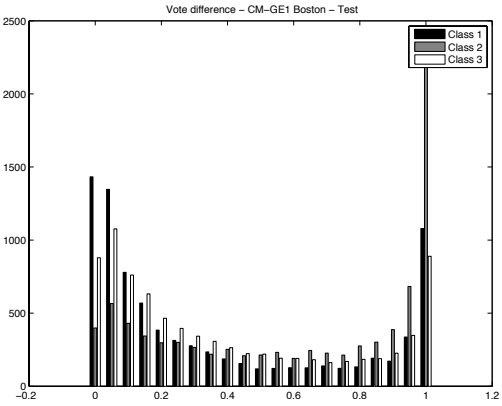
Coverage comparisons are generally categorized into one of three types, each will be discussed with a select example in the following. In each figure, subplots (a) and (b) correspond to CMGE1 train and test, respectively; similarly CMGE2 results are provided in subplots (c) and (d) and PGEC in subplots (e) and (f). For all types, the behavior of the comparison framework (PGEC – Chapter 4, Section 4.1.5) will be demonstrated to be substantially different from the current CMGE models.

The BOST coverage results presented in Figure 8.2 are typical of coverage results returned for the THYD, WINE and WISC problems (Appendix D). In terms of CMGE1 (subplots (a) and (b) for train, test respectively), the majority of the winner / team differences (regardless of class) fall in the extreme bins, approximately 0 or 1, where these correspond to strong support or strong problem decomposition, respectively. Subplots (c) and (d) indicate similar results for CMGE2 with considerably fewer supportive results, irrespective training vs. test scenarios. Class 2 demonstrates the clearest case of this comparison with a large majority of differences being approximately 1. PGEC results on training and test (subplots (e) and (f)) have clearly different distributions from either CMGE model, being approximately symmetric and normal over the range [0,0.95] while having a small response in the final (1.0) bin. These results are highly different from either of the CMGE models and indicate primarily weak or ambiguous responses in PGEC; however, the few cases in the strong difference region indicate some modest potential to make clear distinctions on all three classes. Of the three models, CMGE2 can be seen as making the most strong distinctions between winner and team responses with the fewest moderate and small differences. CMGE1 would be second in this regard, while the inverse is true for PGEC. Of the data sets that demonstrated similar results, THYD revealed perhaps most similarity between CMGE1 and CMGE2, with both models using strong support on the majority class (3) while having the usual distributions on classes 1 and 2. Of the two, CMGE2 again demonstrated more strong distinctive behavior, however with far less regularity than was typically the case for this class of results.

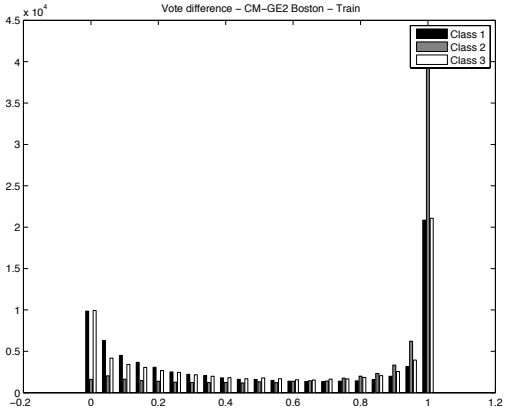
IRIS coverage results are presented in Figure 8.3 and are typical of coverage returned in the IMAG problem (provided in Appendix D). On the whole these results



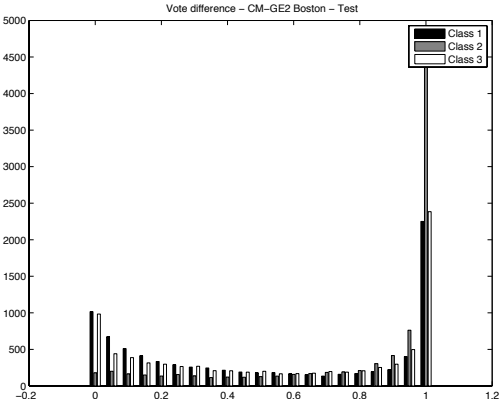
(a) CMGE1 Coverage (Train)



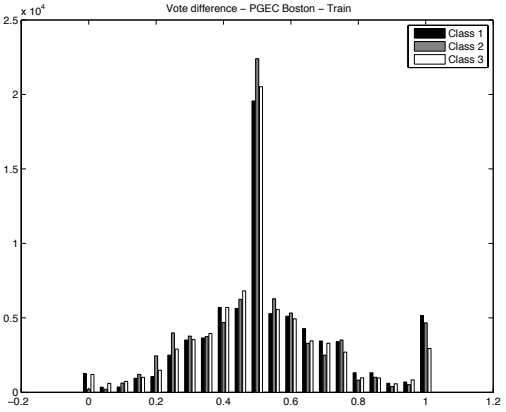
(b) CMGE1 Coverage (Test)



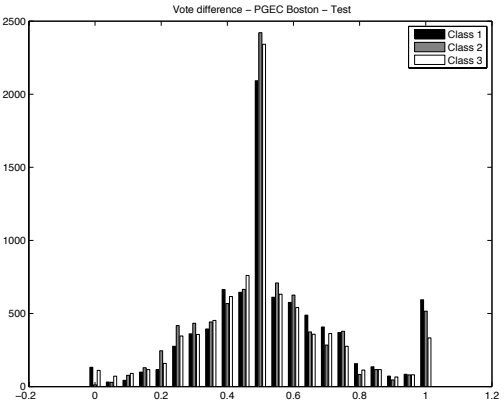
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)

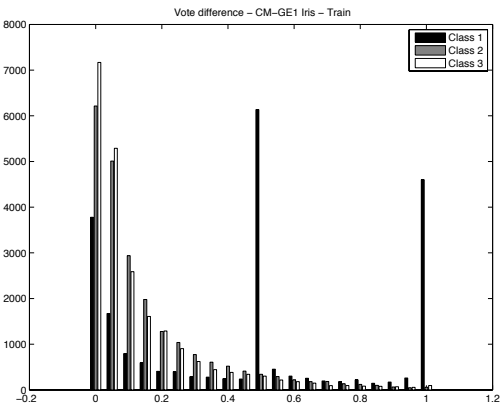


(e) PGEC Coverage (Train)

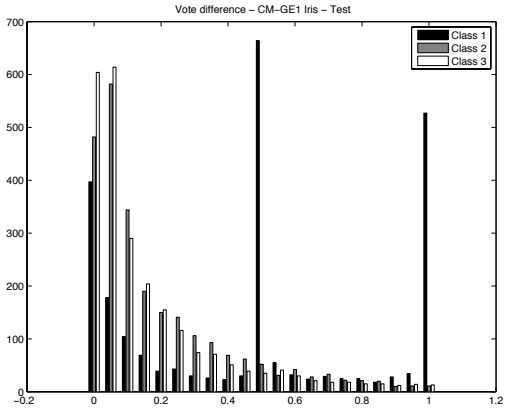


(f) PGEC Coverage (Test)

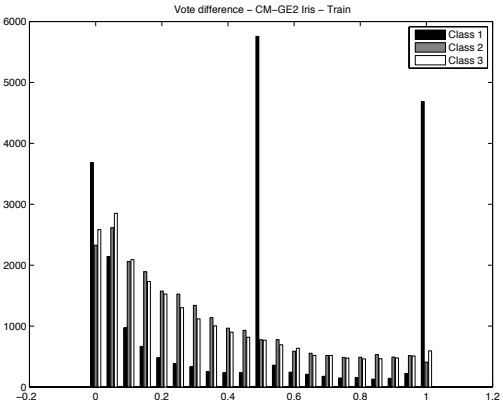
Figure 8.2: Coverage comparison of CMGE 1, 2 and PGEC on BOST.



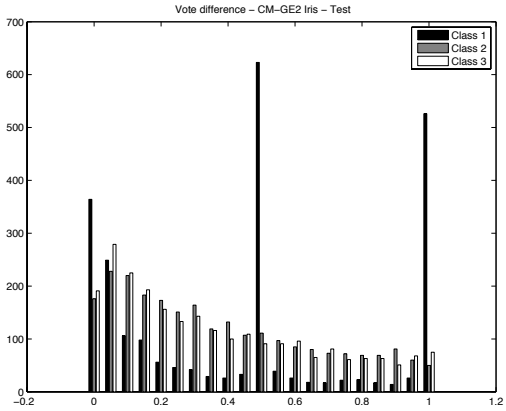
(a) CMGE1 Coverage (Train)



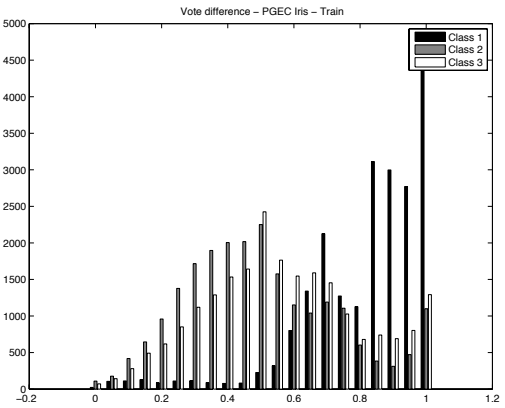
(b) CMGE1 Coverage (Test)



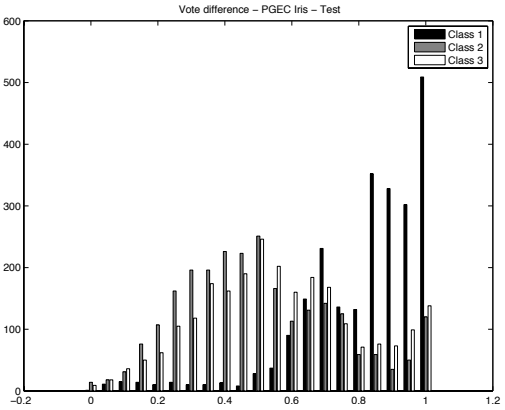
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)



(e) PGEC Coverage (Train)



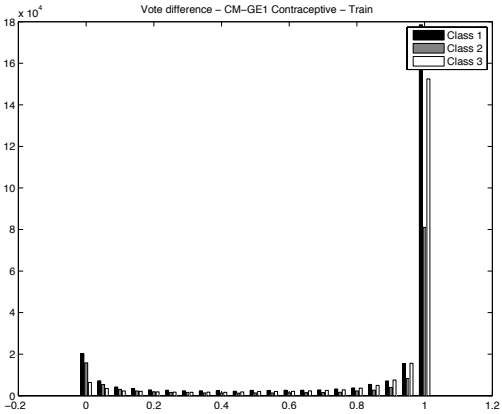
(f) PGEC Coverage (Test)

Figure 8.3: Coverage comparison of CMGE 1, 2 and PGEC on IRIS.

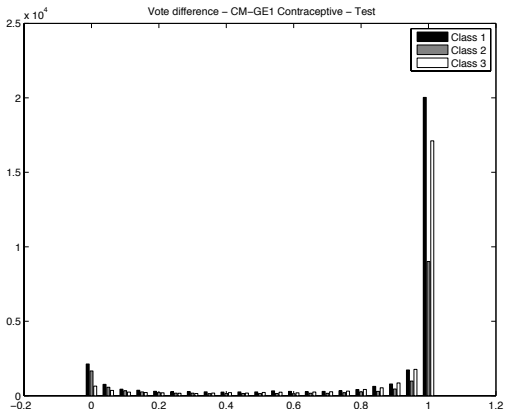
are similar to those of BOST, in that the extreme bins are the most common; however, IRIS has the most results falling near 0 for both CMGE models on the non-linearly separable classes 2 and 3 with most results for class 1 being at either 1 or 0.5. The notable response at 0.5 can be attributed to the median statistic itself; in most instances there are equal number of differences at 0 and 1, making the median difference between the winner and team output equal to the average (0.5) for class 1. Typically only one highly specific solution is required for this class. A similar behavior is observed on class 7 of IMAG. In all other respects, IMAG results are similar to BOST. IRIS also returns the most diverse behavior for the PGEC model, with classes 2 and 3 returning a distribution similar to that of PGEC on BOST; however class 1 behavior is notably different with peaks at 0.7, 0.85, 0.9, 0.95 and 1. These appear to be specific to the IRIS problem, where the linearly separable class 1 can be readily handled with a high-degree of accuracy using a variety of approaches. Of the two CMGE models, CMGE2 employs more moderate to strongly distinctive classifiers on all three classes than CMGE1 and in general both CMGE approaches appear to favor the exclusive use of strong support models over distinctive on classes 2 and 3 as compared with PGEC, which employs weak solutions approximately twice as often as strong, distinctive solutions.

CONT coverage results are presented in Figure 8.4 and are typical of coverage returned in the BUPA and PIMA problems (provided in Appendix D). Of the coverage results compiled, these correspond to the more difficult classification problems and share the preference for strong differences (particularly in test) between CMGE1 and CMGE2, where this is more pronounced in the CMGE2 model on BUPA and PIMA while both are primarily in the 1.0 bin on the CONT problem of Figure 8.4, subplots (a) (b) (c) and (d). Distributions on each of these problems are consistent over train and test within each approach. In all cases, PGEC (subplots (e) and (f)) again employs the weak approach involving moderate differences between winner and team outputs.

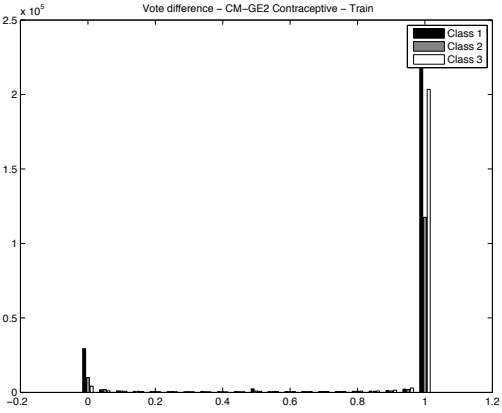
These results indicate a clearly different approach to classification problem decomposition with the CMGE models typically representing stronger, decisive decompositions or strong supporting classification as opposed to the moderate and ambiguous



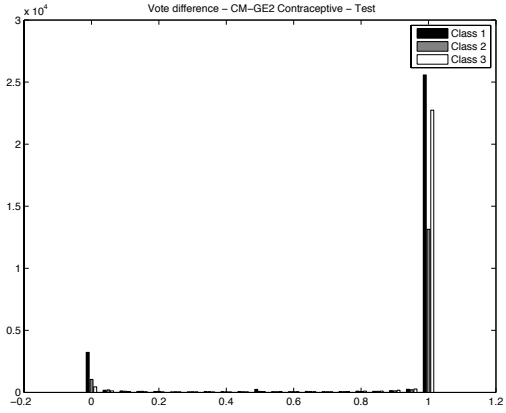
(a) CMGE1 Coverage (Train)



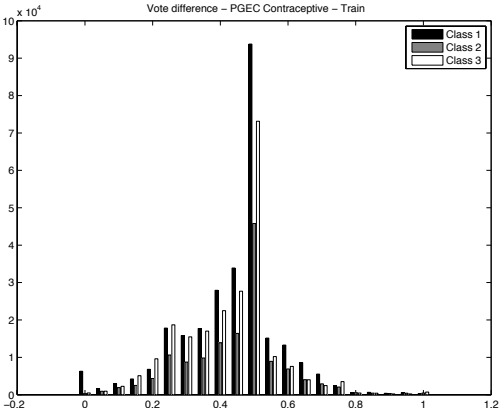
(b) CMGE1 Coverage (Test)



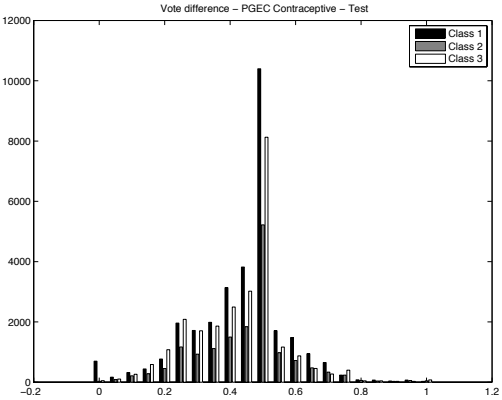
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)



(e) PGEC Coverage (Train)



(f) PGEC Coverage (Test)

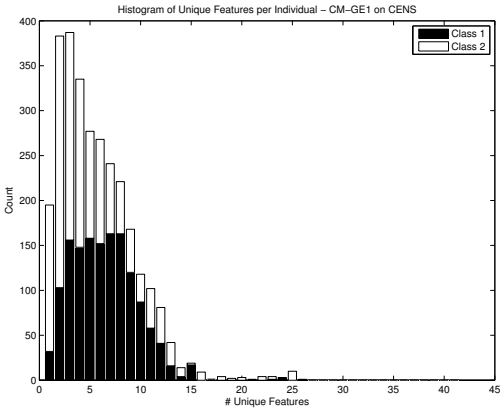
Figure 8.4: Coverage comparison of CMGE 1, 2 and PGEC on CONT.

differences observed in all problems for the PGEC approach. The extent of this preference varies between problems and CMGE model employed. CMGE1 makes more intermittent use of strong support or moderate differences on the straightforward problems (e.g., WISC and WINE), where large differences are frequently relied upon by CMGE2.

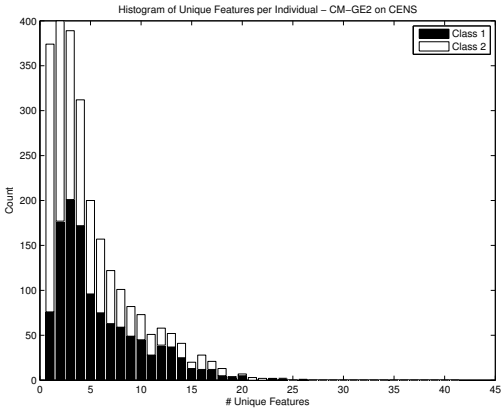
8.3 Feature Selection

The CMGE framework performs automatic selection of discriminating features, decreasing evaluation overhead and solution complexity. This section will highlight two representative problems (CENS in Figure 8.5 and PIMA in Figure 8.6) in terms of number of unique features appearing in solutions and number of occurrences of each feature. Counts of unique features per individual are provided in subplots (a) and (b) for CMGE1 and CMGE2, respectively. Feature occurrences are examined from two perspectives: as solution appearance counts in subplots (c) and (d), and as total counts in subplots (e) and (f), for CMGE1 and CMGE2 models respectively. The key difference between feature occurrence counts provided in each figure is that feature repetition (within a solution) is not captured by individual occurrence plots, whereas this property is retained in total occurrence plots. All plots are presented as class-wise stacked histograms. Full feature selection results are provided in Appendix E.

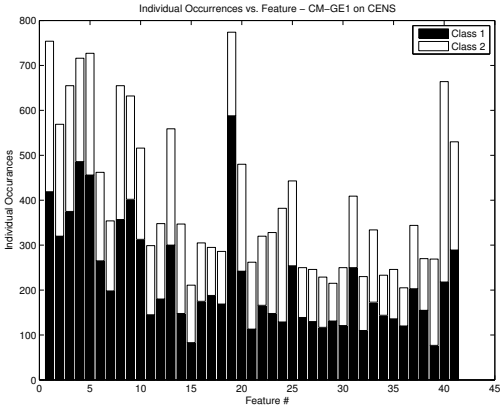
Figure 8.5 provides the results of the CENS feature analysis. These results are typical of several problems in the low number of unique features involved in most solutions and a high degree of selectivity in terms of the choice of features actually employed in the solutions. Comparing subplots (a) and (b), it is clear that the two CMGE models have different distributions of feature counts for the CENS problem; specifically CMGE1 results of subplot (a) indicate considerably fewer solutions with 3 or fewer unique features than CMGE2 results of subplot (b); however, the CMGE1 results are noticeably more concentrated between 1 and 10 features, while CMGE2 results have slightly more prominent tail extending into higher feature usage. This effect is more apparent in the class 1 results on CENS however this can be observed on several other problems, including IMAG, KD99 and THYD. In terms of feature preference, both models demonstrate a clear bias toward features 5, 13, and 19. CMGE1



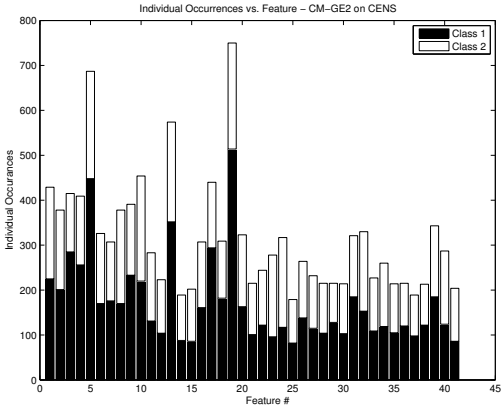
(a) CMGE1 Features per Individual



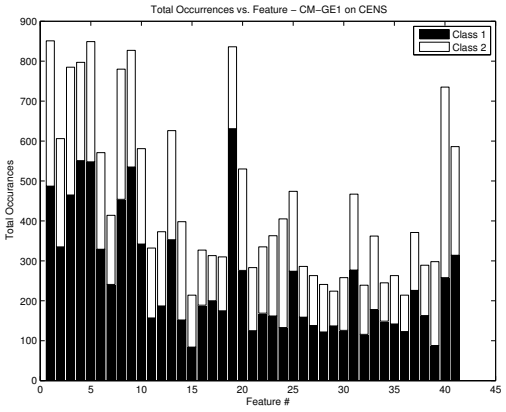
(b) CMGE2 Features per Individual



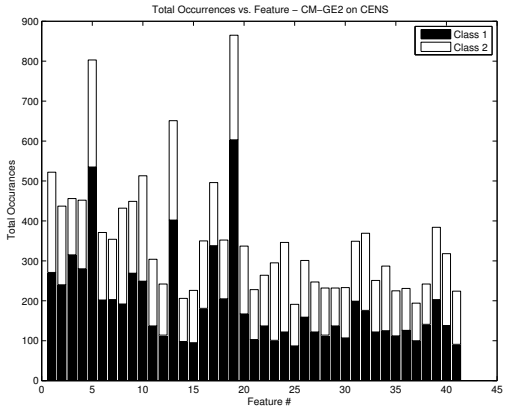
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

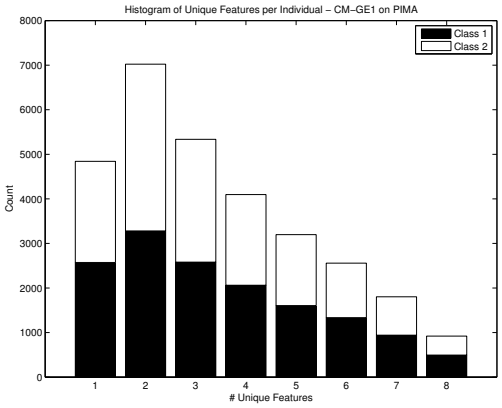


(e) CMGE1 Total Occurrences

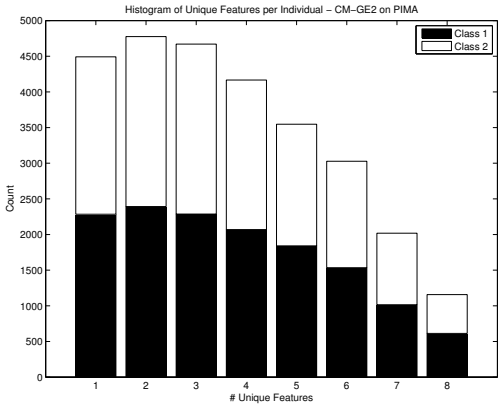


(f) CMGE2 Total Occurrences

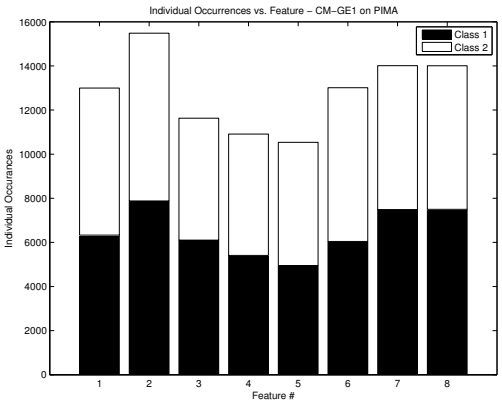
Figure 8.5: Feature Analysis: CMGE 1, 2 on CENS.



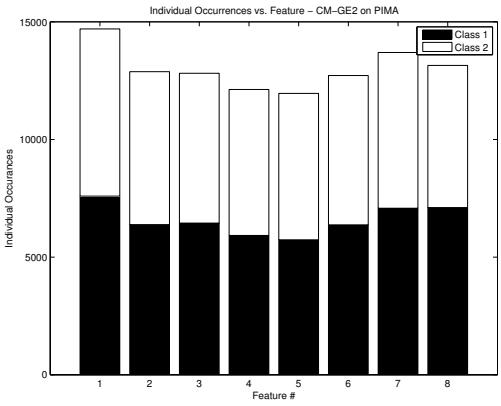
(a) CMGE1 Features per Individual



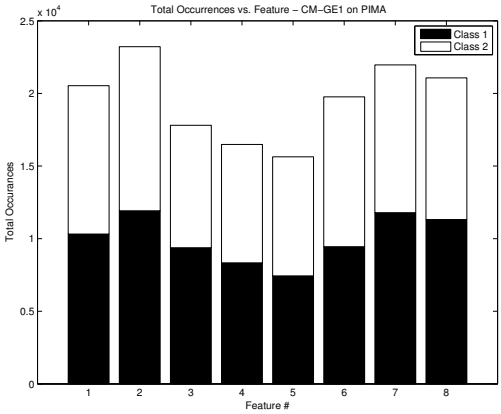
(b) CMGE2 Features per Individual



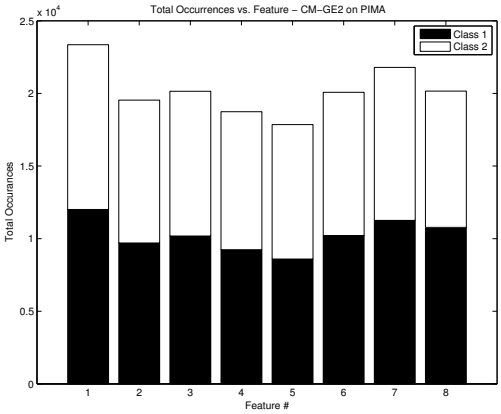
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences



(e) CMGE1 Total Occurrences



(f) CMGE2 Total Occurrences

Figure 8.6: Feature Analysis: CMGE 1, 2 on PIMA.

results of subplots (c) and (e) additionally identify features 1-4, 8, 40, and 41, with the latter two also being specific to class 2. The feature selection behavior is also readily apparent on BOST, THYD, KD99, SHUT and WINE where between 3 and 5 features are obviously preferred by solutions while several features remain noticeably underrepresented in each of these problems. The total occurrence histogram of subplot (e) appears to indicate more repetition of features in CMGE1 than CMGE2 (subplot (f)), and this additionally appears to be the case for the IMAGE, SHUT and WISC problems, with all others being generally indistinguishable in this respect.

Figure 8.6 presents the results of the PIMA problem which are representative of a second type of result that is typical of most other problems (including BOST, CONT, IRIS, SHUT, WINE and WISC) in terms of high degrees of similarity between feature usage distributions returned by CMGE1 and CMGE2 in subplots (a) and (b), respecitvely. Despite the visual differences between these histograms due to disparities in scale of y-axes, both CMGE1 and CMGE2 return highly similar results, with both indicating only marginal preference for using fewer unique features. Moreover, the inability of either CMGE to select specific features (as indicated in subplots (c) and (d) for CMGE1 and CMGE2, respectively), is consistent with results returned by other difficult problems, including BUPA and CONT.

8.4 Summary

The PGEC methodology, from which CMGE borrows, integrates RssGE with a competitive coevolutionary model for retaining the most appropriate learners and discriminatory test points during training (due to de Jong's IPCA style algorithms, Algorithm 25, Section 3.8). As such, both models result in solutions that take the form of a Pareto front of learners. PGEC is configured to recombine labels using a majority vote under unseen data. Conversely, CMGE has a winner-take-all basis, reflecting the explicit problem decomposition approach to performing the feature space to class label mapping. In this chapter we provide empirical evidence to support these assumptions. The PGEC model is shown to take a classical ensemble / weak learner approach to team building; whereas the CMGE model generally produces classifiers that respond to unique subsets of the data set. In effect, we have validated that

the explicitly cooperative mechanism enforced by objective three of CMGE (Section 3.7.4) in combination with the local membership function, does provide an effective mechanism for problem decomposition.

We conclude this chapter by verifying the bias of EC to select samples of features where appropriate, although on the more difficult problem domains all features may well be utilized.

Chapter 9

Archive Size Parameter Analysis

Results presented in this chapter pertain to experiment E12 (Table 4.6) and examine the performance of the CMGE1 classification framework as applied to three of the benchmark data sets over a systematic range of learner and point archive capacities. These results establish a general set of recommendations for parameterization of the framework and act as a baseline for future studies on the significance of archive size and maintenance through pruning. All results are provided as surface plots of classification performance (z-axis) as a function of the learner archive (x-axis) and point archive (y-axis) sizes. Archive sizes assume values of $S = \{10, 20, 30, 40, 50\}$ and the surface is defined over parameterizations $P = S \times S$. Each of the 16 surface tiles are shaded according to their associated scales to indicate surface height on the z-axis, with larger values (corresponding to lighter shades) being desirable in all statistics aside from false positive rate.

Overall results in terms of accuracy and score will be provided for each of the three benchmark data sets (BOST, CONT, THYD) in Figures 9.1, 9.2, 9.3, respectively in Section 9.1. Within each figure, subplots (a) and (b) provide accuracy results while (c) and (d) provide score results for train and test, respectively. Finally, Section 9.2 presents a class-wise sample of detection and false positive rate results on the three class BOST problem in Figures 9.4 - 9.6. All remaining class-wise results are provided in Appendix F.

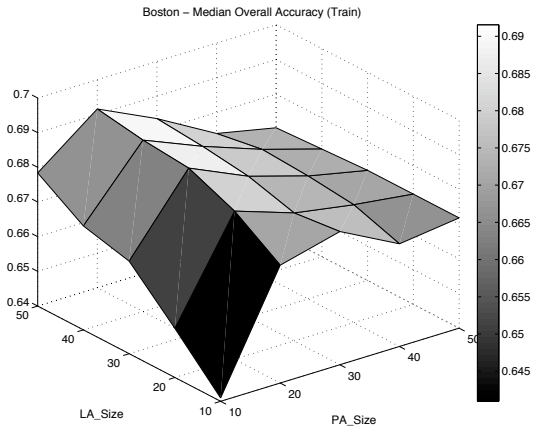
9.1 Overall Results

Overall results in terms of accuracy and score illustrate that the combination of small learner and point archives capacities (≤ 10) are typically much less effective than any other parameterization, irrespective of the data set. Moreover, the small archive configuration consistently demonstrates an approximately minimal performance on

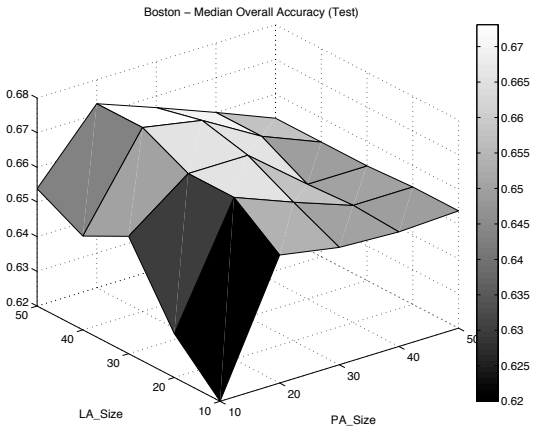
both training and test data, which indicates that such an economical choice of parameters is associated with a general failure to support learning. Moving along either axis while fixing one parameter at 10 coincides with overall improvement, with this being initially more substantial when associated with an increase in the maximum point archive size. This is indicated by a lighter shading of the tiles adjacent to the origin (10,10) in the surface plots for BOST (Figure 9.1) and CONT (Figure 9.2), with a steeper slope when moving from (10,10) to (10, 20) as opposed to (20,10). An apparent deviation from this trend exists in the overall results of THYD. While improvement again appears associated with an increase in either parameter, it is unclear that the point archive size is more influential in this regard. This is particularly clear in the surface of THYD test score in subplot (d), Figure 9.3 where additional learner archive capacity appears to initially support better generalization.

The overall results appear to support a preference for the selection of a moderate as opposed to either conservative or overly capacious point archive sizes, whereas the maximum learner archive size (while generally associated with marginal improvement) appears to have less impact on the overall classification performance, beyond a default setting of 10. This is most clearly illustrated by the score surfaces in subplots (c) and (d) of the BOST and CONT data sets of Figures 9.1 and 9.2, respectively, where a ridge of strong results are returned at the point archive setting of 20. This moderate point archive size also appears to be most strongly associated with improvements based on increasing the learner archive size, whereas larger point archive sizes lead to a plateau or outright deterioration in performance. This plateau effect is illustrated in the BOST score test results of Figure 9.1, subplot (d), with deterioration in overall performance observed with increased point archive sizes in subplots (a), (b) and (c).

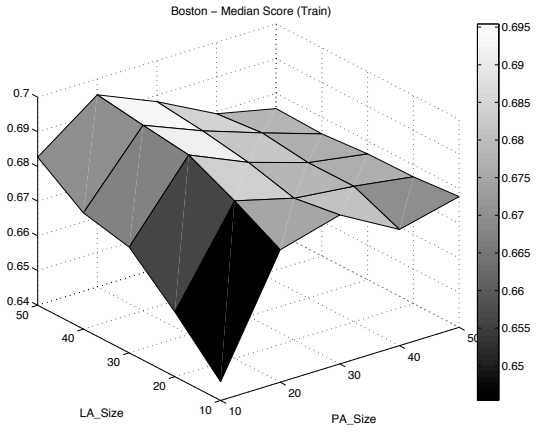
Large choices for both learner and point archive capacities do not appear to be strongly associated with peaks in the overall performance metrics in either the training or test scenarios, with this being particularly evident in the score surfaces. Despite the typically good performance and stability of the surface region around the maximum parameterizations in terms of overall accuracy, these settings are rarely optimal as will be illustrated by the class-wise BOST results the follow in Section 9.2. Moreover, the stability in overall accuracy that may be achieved through naïve selection of



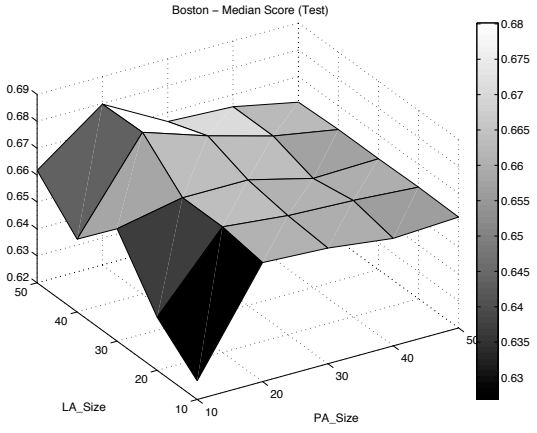
(a) Median Overall Accuracy (Train)



(b) Median Overall Accuracy (Test)

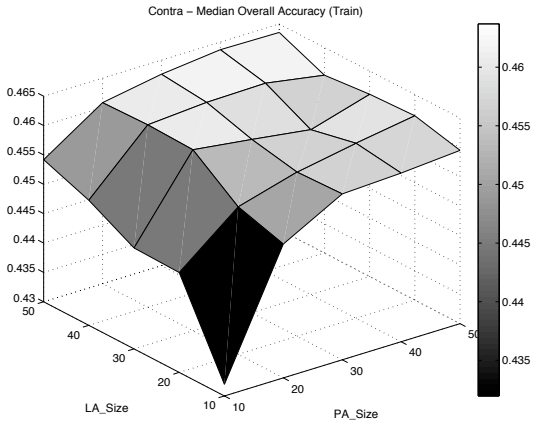


(c) Median Score (Train)

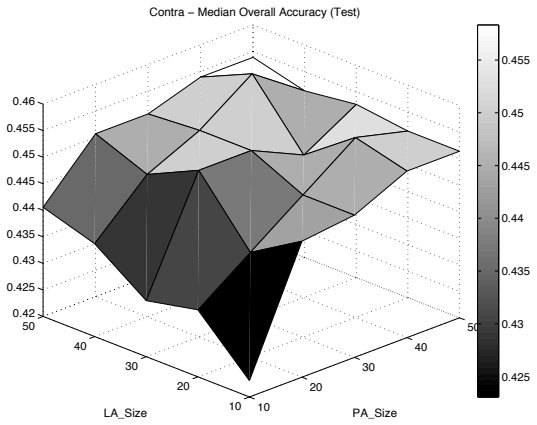


(d) Median Score (Test)

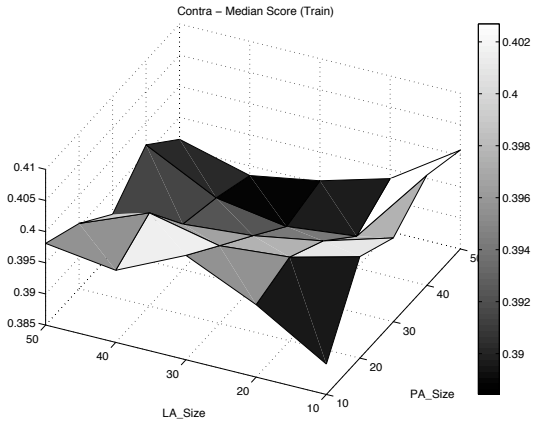
Figure 9.1: Parameter analysis of CMGE 1 median Overall Accuracy (a) (b) and Score (c) (d) on BOST.



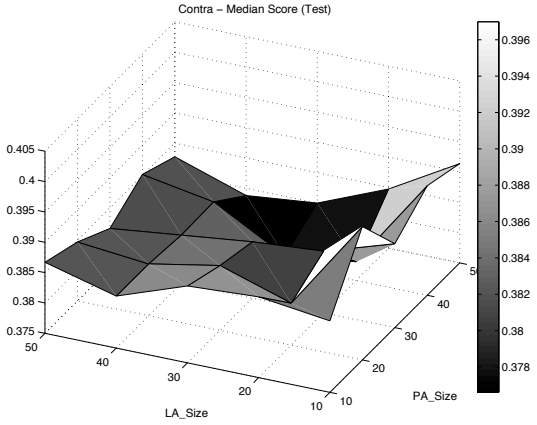
(a) Median Overall Accuracy (Train)



(b) Median Overall Accuracy (Test)

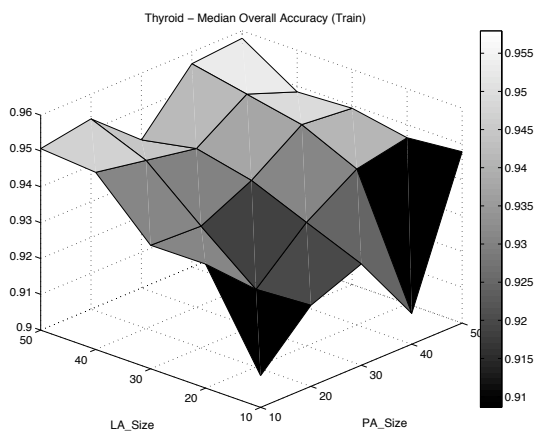


(c) Median Score (Train)

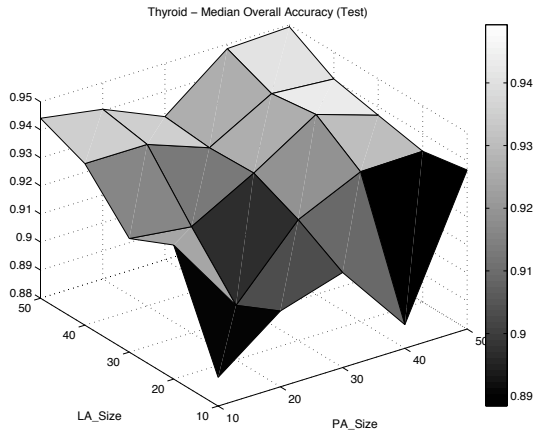


(d) Median Score (Test)

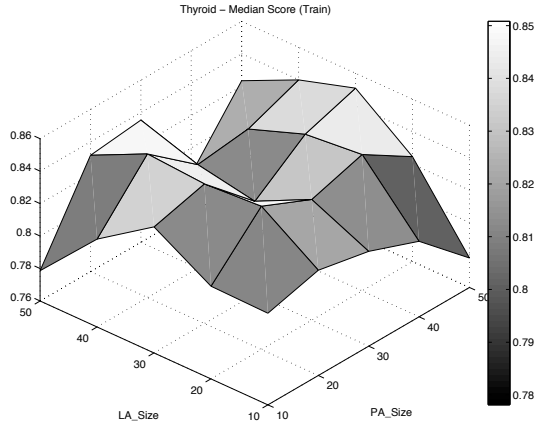
Figure 9.2: Parameter analysis of CMGE 1 median Overall Accuracy (a) (b) and Score (c) (d) on CONT.



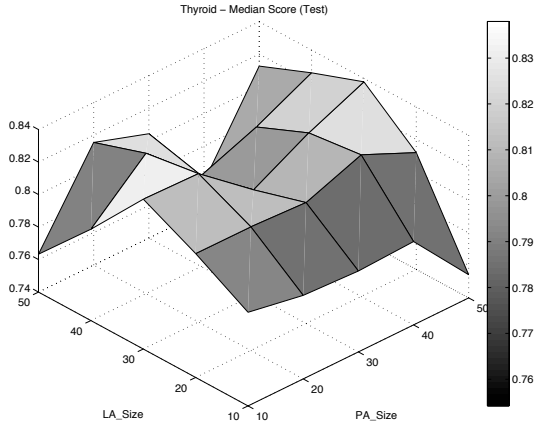
(a) Median Overall Accuracy (Train)



(b) Median Overall Accuracy (Test)



(c) Median Score (Train)



(d) Median Score (Test)

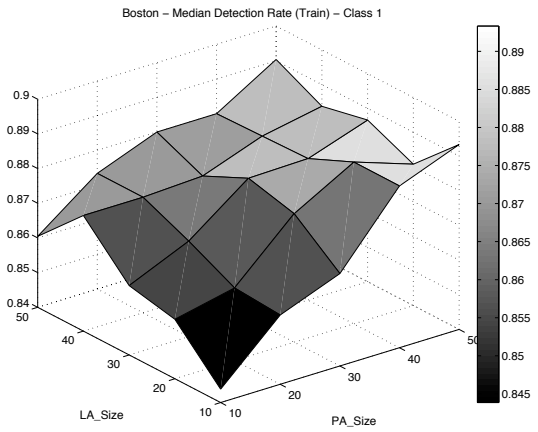
Figure 9.3: Parameter analysis of CMGE 1 median Overall Accuracy (a) (b) and Score (c) (d) on THYD.

large archive size parameters will frequently be offset by an increase in performance overhead. Clearly the optimal settings for overall performance are problem specific; however, a straightforward recommendation is for modest point archive capacity (20-40) with as much allocation to learner archive capacity as can be tolerated under performance constraints (≥ 30).

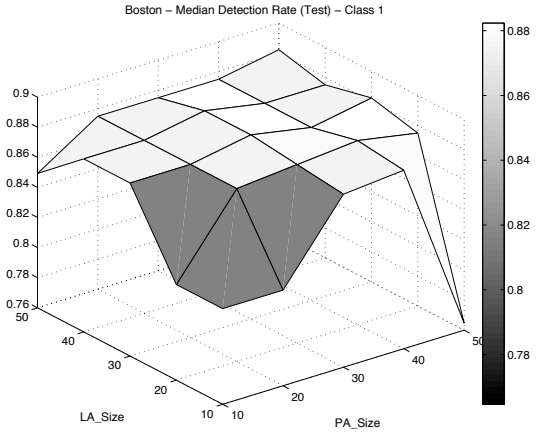
9.2 Class-wise Results

Class-wise surface plots are provided for detection rates in subplots (a) and (b) and false positive rates in subplots (c) and (d) for training and test, respectively. Classes 1 and 3 (Figures 9.4 and 9.6) of the BOST problem demonstrate similar and highly stable detection rate results across the various parameterizations, as indicated by subplots (a) and (b) in each figure. The training results of both classes exhibit improvement in detection rates with increased archive sizes; moreover, point archive size appears to effect this result to a greater extent than the learner archive size, as readily demonstrated by subplot (a) of Figure 9.4. Detection rates on test for both classes plateau when a parameterization other than (10, 10) is employed, with some sporadic results being observed at the extreme point archive size selections (50). Class 2 results for detection rate are given in Figure 9.5, subplots (a) and (b) (train and test, respectively) and illustrate the optimal combinations of point and learner archive sizes at the point archive settings of 20 and 30. Moreover the larger learner archive sizes appear to support better training results, as indicated in subplot (a), but it is not clear that this improvement carries over to the test results of subplot (b).

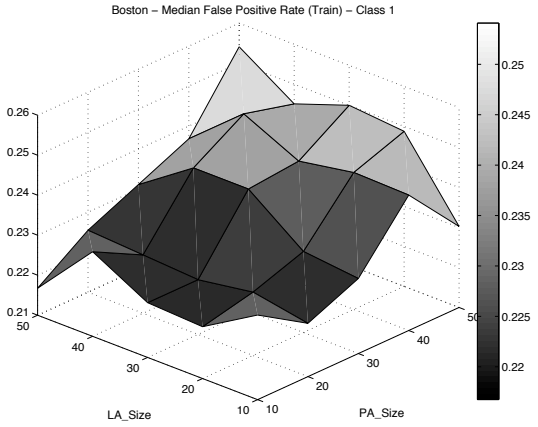
Subplots (c) and (d) of Figures 9.4 and 9.6 demonstrate that false positive rates of class 1 and 3 results are again largely similar, both in terms of training and test results, while the same subplots in Figure 9.5 illustrate the classical trade-off of detection rate at the expense of false positive rate with little appreciable variation in the false positive rate ($\sim 3\%$) in response to changes in the archive size parameters. In both training cases of classes 1 and 3, indicated in subplot (c) of each Figures 9.4 and 9.6, low false positive rates are returned under the large learner archive ($\sim 40 - 50$) and moderate point archive size ($\sim 20 - 30$) combinations. Moreover, large selections



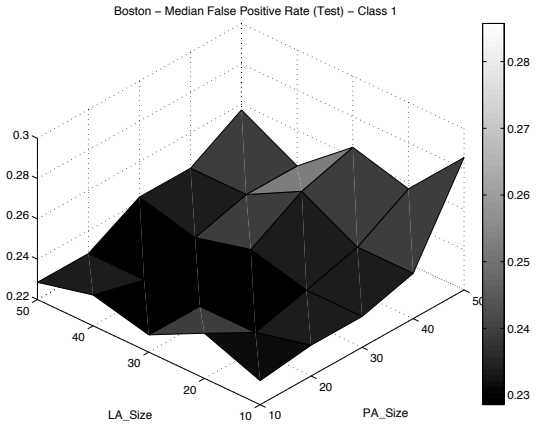
(a) Median DR (Class 1 - Train)



(b) Median DR (Class 1 - Test)

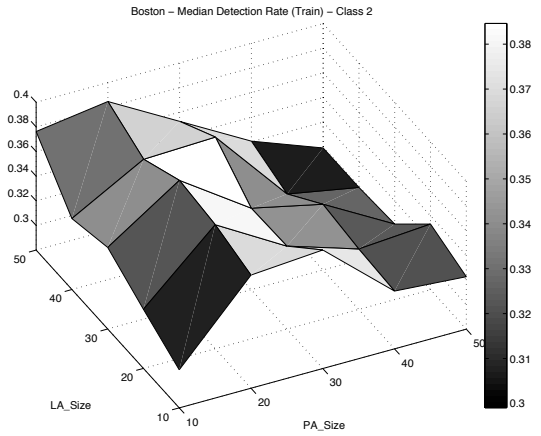


(c) Median FPR (Class 1 - Train)

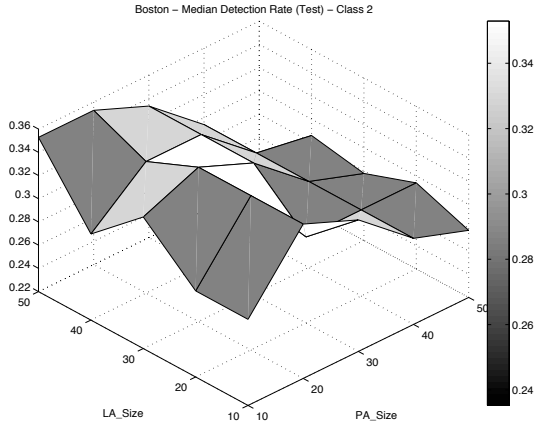


(d) Median FPR (Class 1 - Test)

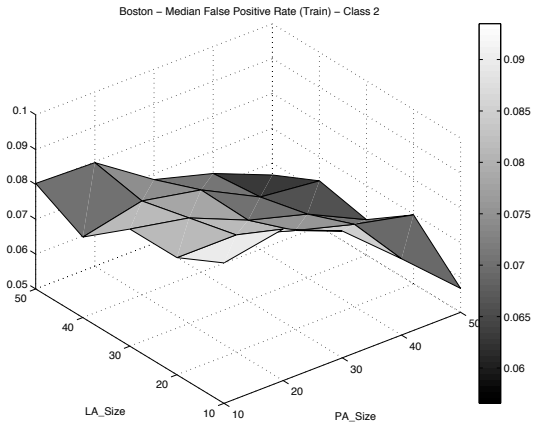
Figure 9.4: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on BOST, class 1.



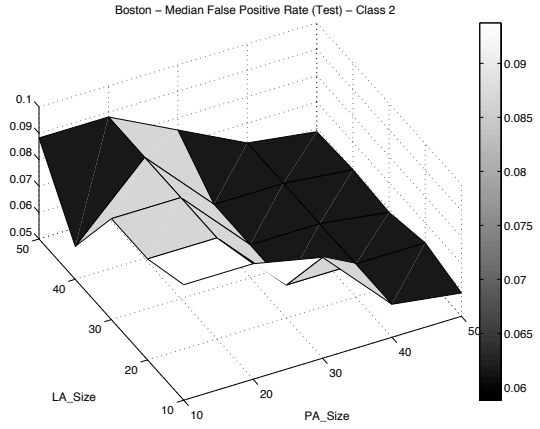
(a) Median DR (Class 2 - Train)



(b) Median DR (Class 2 - Test)

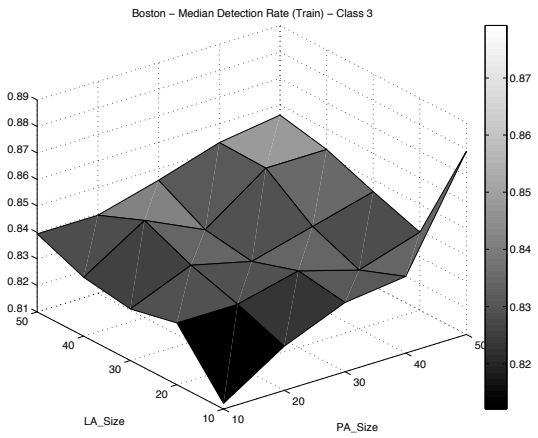


(c) Median FPR (Class 2 - Train)

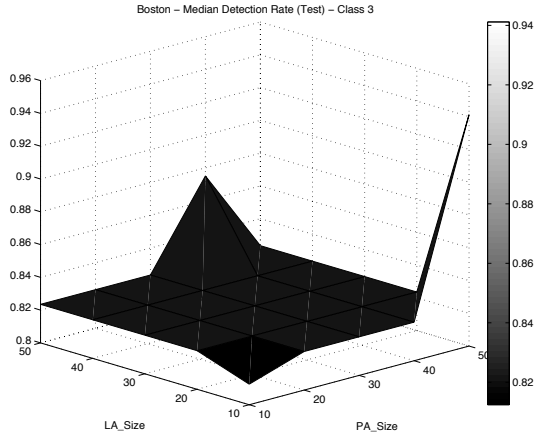


(d) Median FPR (Class 2 - Test)

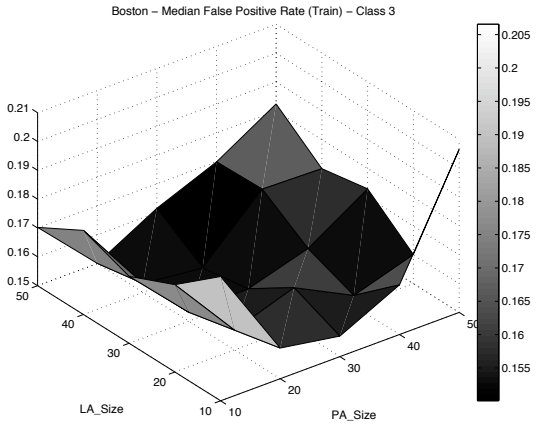
Figure 9.5: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on BOST, class 2.



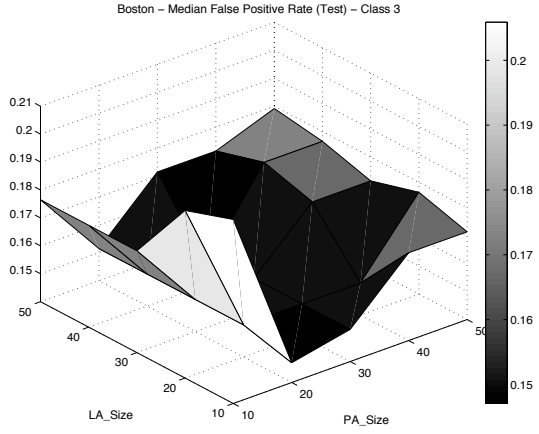
(a) Median DR (Class 3 - Train)



(b) Median DR (Class 3 - Test)



(c) Median FPR (Class 3 - Train)



(d) Median FPR (Class 3 - Test)

Figure 9.6: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on BOST, class 3.

for both parameters resulted in the approximate worst case (highest false positive rates) in training on both classes. Test results of subplot (d) in each figure provide additional evidence to support the choice of moderate point archive sizes, with both classes having increased false positive rates corresponding to increased point archive sizes. Moreover, given that detection rates on both classes remain unchanged in non-default parameterizations, a dominant result occurs when the minimum false positive rates are reached in association with moderate point archive sizes. A clear preference is therefore shown for moderate point archive sizes in this problem.

9.3 Summary

Beyond the selection of small archive limits for either learner or point archives, the CMGE algorithm appears to function adequately with capacity for twenty to forty exemplars / learners. No real benefit appears to be gained by allowing for larger learner (point) archives relative to point (learner) archives.

Chapter 10

Conclusion

The final chapter begins with a synopsis of the key ideas put forth in the development of this thesis. We then proceed with a reiteration of our original objectives, emphasizing how each has been demonstrated by our experiments and addressed by the proposed classification framework, providing a high-level review of supporting results where appropriate. This is followed by a discussion of our main contributions in terms of the framework and its development (i.e., the novel aspects of the algorithmic design) and a short commentary on the implications for classification where we restate the case for a scalable GP-based approach that enables problem decomposition. The thesis concludes with a short set of recommendations for future work.

10.1 Synopsis

A novel framework for multi-class classification under the Genetic Programming context has been introduced. Two variants of the framework (dubbed CMGE1 and CMGE2) have been implemented and studied over a diverse, twelve problem benchmark of real-world data sets representing both binary and multi-class problems. The proposed framework permits a GP approach to classification with large and unbalanced data sets where the traditional issues of scalability, problem decomposition and solution transparency are addressed simultaneously within a coevolutionary multi-objective training algorithm.

This thesis began by introducing Machine Learning and the classification task, focusing on the essential ML design considerations, including: specification of representation; cost function definition (and goal identification); and approach to credit assignment. The case was made for flexibility in framework design in terms of the specification of representational elements and definition of cost function. The merits and tradeoffs associated with credit assignment as it relates to stochastic search

are discussed with its exploration potential combined with mechanisms for avoidance of local optima highlighted as desirable qualities in a learning framework. We next introduce the Genetic Programming paradigm of Evolutionary Computation, with emphasis on the flexible design characteristics inherent in the approach, including user specification of representation and cost (fitness) function. Next, the classification context of GP was introduced and the classical drawbacks were discussed in detail. Finally our research objectives were defined; specifically, we set out to address the issues that have traditionally precluded GP approaches to classification: scalability, class imbalance, solution transparency, problem decomposition and multi-class applications.

Chapter 2 endeavored to establish the relevant background material and work related to the current research objectives. A review of the Grammatical Evolution variant of GP was provided; however, we reiterate that the contributions of this thesis are independent of the formulation of GP under consideration. The GE review was carried out with special attention to the basic ML design elements, including a discussion of the highly flexible grammatical (CFG-based) approach to representation and our design considerations for context sensitive crossover and terminal specific mutation operators, citing research indicating their potential to aid in the credit assignment process. We proceeded to survey material from the GP binary and multi-class classification literature, making the case for a 'novelty detection' approach that employs a coevolutionary multi-objective training algorithm. Our literature review highlighted background material relevant to developing evolutionary multi-objective optimization and we surveyed recent work relating EMO approaches to the GP context. The literature considered relevant to the problem decomposition and scalability objectives was finally presented and we made the case for a balanced competitive coevolutionary approach to training.

Chapter 3 presented a detailed account of the algorithms employed by the proposed framework and the associated computational complexities. We proceeded to provide pseudo-code-level descriptions of all algorithms noting design decisions that were considered critical to addressing our original objectives, including: use of a local

(Gaussian) membership function to establish the 'novelty detection' model of operation; an EMO design allowing for strong (cooperative) problem decomposition along with low error and solution transparency; multi-class competitive coevolution using a balanced sampling heuristic to address scalability and class imbalance, respectively; class-wise early stopping criteria; and finally, a post-training winner-take-all voting policy.

Chapter 4 outlined our benchmarking methodology including eight comparative algorithms (four GPs: Canonical GE, StdGE, RssGE, and PGEC; two Neural Network architectures: LP and MLP, and three deterministic classification algorithms: 1R, NB and J48 (the C4.5 release 8 implementation)) that were to be compared in terms of classification performance on twelve real-world classification problems. The approach to evaluation was presented and a statistical evaluation framework for comparing stochastic algorithms (including classification performance, training time and solution complexity[1]) was provided. Moreover, we proposed a framework for making comparisons between the GP and deterministic models which addresses a key question when comparing such algorithms, namely: how likely the stochastic model is to perform equally or better as a result of the ability to make alternative decisions during model building.

In Chapter 5, comparisons were made to the standard canonical GP approach in terms of classification performance, training time and solution complexity indicating that the CMGE models provided statistically significant improvements across the board. Next the baseline models (StdGE and RssGE) were compared with the proposed approach, demonstrating that RssGE returned the more competitive results of the two; however, the results of the CMGE models were clearly demonstrated as superior under our evaluation framework. Moreover, the gains in classification performance were achieved with vastly lower computational overhead (as compared with the canonical model) while solution size was shown to be adequately contained per individual. Further comparative analyses were carried out in Chapter 6 with respect to two feed-forward Neural Network architectures (Linear Perceptron and Multi Layer Perceptron models) that were trained using a robust second order algorithm. The

[1]Clearly only the classification performance is directly comparable in the case of the Neural Network comparisons

CMGE models returned consistent and largely competitive results, providing statistically significant improvements over the Neural Network approaches in certain cases; however, on the whole, algorithm preferences in terms of classification performance were shown to vary from data set to data set. Results and analyses of the deterministic experiments were provided in Chapter 6, which revealed that the proposed approach is capable of returning highly competitive classifiers (indeed many results represent improvements in classification performance) with respect to the standard ML classifiers considered. These comparative results were very encouraging, particularly considering the fact that no attempts were made at optimizing the CMGE framework parameters and no special preprocessing of the data was required.

Chapter 8 demonstrated a distinctive style of problem decomposition achieved by our 'novelty detection' approach in terms of voting behaviors as compared to those due to a discrimination-type wrapper function employed in the 'weak learning', binary coevolutionary GP configuration (i.e., Lemczyk's PGPC algorithm [70]). These intra-class comparative results provided additional evidence to the differing voting behaviors first observed in the inter-class comparisons (in terms of classification results) carried out in Chapter 5, where the PGEC algorithm was noted to provide mainly degenerate solutions. These were frequently observed to be due to incompatibly of teams between classes (initializations) under the default majority voting policy.

In Chapter 9 we confirmed that an appropriate selection of the archive size parameters may have the potential to effect further gains in classification performance, while noting that larger choices (with the capacity to negatively influence computational overhead) do not necessarily lead to improved classification results under our framework. Finally, we provided evidence for feature selection under the CMGE models. Such behavior has the potential to provoke further insight to solutions provided; however, we noted that this result was not necessarily consistent over all data sets.

10.2 Objectives Realized

In the introduction of this thesis, five objectives for our research were outlined specifically aimed at confronting the problems that frequently preclude a GP approach to classification. These will be discussed in the following sections with emphasis on

how each has been demonstrated and subsequently addressed by our framework, citing specific design decisions and results relevant to establishing our success on each objective.

10.2.1 Scalability

Without recourse to hardware specific speedups, the canonical form of GP does not explicitly address the prohibitive computational overhead associated with problems involving large numbers of training exemplars (tens of thousands or millions; see Equation 1.2). This performance issue is, in large part, tied to individual evaluations over all training exemplars in the costly inner-loop of GP, defining an undesirable performance characteristic that is well documented in the literature [108] [22] [6]. We have discussed the grounds for the scalability issue at length in Chapters 1 and 2 and demonstrated this classical pathology in Chapter 5, where a modestly configured canonical GP on modern hardware typically required 39 to 75 hours per initialization on the CENS data set.

The scalability objective was realized in two ways. First, through a reformulation of the Incremental Pareto Coevolution Archive (IPCA) algorithm [23] that allows the GP learners to train based on a balanced (in versus out-of-class) sample of training data combined with the most 'useful' points for learning in each class. Second, class-wise early stopping criteria allows for dynamic and efficient reallocation of GP resources (population members) as each class converges. The scalability of the CMGE framework is established by the computational complexity analysis outlined in Chapter 3 which suggests that the proposed framework will provide a speedup over the canonical approach assuming archive and population sizes $S \leq \sqrt[3]{|TD|}$. This analysis is readily supported by empirical results (in terms of training time) provided in Chapter 5 where we confirm that the training process under the proposed framework is reduced to minutes or seconds per initialization while returning overall classification results that represent improvements that are statistically significant as compared to the baseline GP models over the vast majority of problems.

10.2.2 Class Imbalance

We have demonstrated the tendency of poor representation of minority classes in the training set to lead GP individuals to focus on the majority classes in an attempt to maximize fitness with the canonical results provided for unbalanced problems in Chapter 5. Moreover, a cursory review of the StdGE class-wise detection results (provided in Appendix G) clearly demonstrates this pathology under the BUPA, CENS, CONT, KD99, PIMA, SHUT and THYD problems where consistent detection rates are provided for the majority classes only. Drawing on the research of Weiss and Provost [116], the framework developed in this thesis employs a balanced view of each class as well as balanced (in versus out-of-class) class-specific archives to directly address this shortcoming. In addition to returning the best overall training and test results on 10 of the 12 data sets in comparison to the baseline and PGEC GP approaches (Chapter 5), the CMGE framework also provided consistently competitive results with the three deterministic classifiers (Chapter 7) as well as the two Neural Network architectures (Chapter 6). These results are clearly confirmed by the class-wise detection and false positive rates supplied in Appendix G.

10.2.3 Solution Transparency

The solution transparency issue is well established in the literature and involves the potential for code growth or 'bloat' with longer evolutionary runs of GP, directly impacting the simplicity and understandability of solutions [6] [61]. In light of earlier results due to [25] [93], we adopt a multi-objective evaluation scheme in the learning cycle that employs an explicit parsimony objective (described in Chapter 3) that specifically addresses this growth characteristic by simultaneously encouraging low error, short expression lengths and strong coverage among individuals during evolution. Empirical evidence of this classical pathology was demonstrated in the canonical GE results (particularly for the THYD problem) provided in Chapter 5. Experimental results under the proposed framework indicate that our design indeed discouraged solution complexity and variation was typically well contained per individual (especially in CMGE2) in comparison to the canonical and baseline models. Moreover, no

baseline model was able to provide significantly lower complexity despite being explicitly limited in evaluations. Further evidence for solution transparency is provided by the feature selection results presented in Chapter 8, where the CMGE models were confirmed to prefer certain features over others and fewer rather than more unique features on many problems although we acknowledge that this behavior was notably absent on some of the more difficult problems (e.g., PIMA, BUPA) and/or those involving very few features (e.g., IRIS).

10.2.4 Problem Decomposition

As discussed at length in Chapter 2, problem decomposition in the GP context has typically referred to evolution of explicitly reusable modules or 'building blocks' (e.g., ADFs) to enable efficient code use and exploitation of learned components. In this thesis automatic problem decomposition is developed such that multiple individuals interact (both between classes and within classes) to effectively solve separate sub problems cooperatively, in contrast to the single 'super individual' approach of the canonical model which assumes that a single individual is sufficient and/or appropriate to solve a problem. This objective is addressed by two design properties of the proposed framework. Firstly, the use of a local membership function (in the form of a Gaussian) enables individuals to act as novelty detectors rather than discriminators. Introduced in in Chapter 3, the LMF is employed to enforce class-consistent cluster mappings, where correct assignment of local points to the individual cluster is rewarded without explicitly requiring that all points be handled by any given mapping. Under this design, multiple individuals are permitted to evolve and contribute to the final solution, each having potentially differing significance in terms of the decomposition of the overall problem. Secondly, when designing the EMO component of the model (discussed in Chapter 3), care is taken to construct the objectives such that collaborative behavior with respect to other members of the corresponding (class-specific) Pareto archive is explicitly encouraged; that is to say, individuals are rewarded for classifying points that are not already classified correctly by potential team members (individuals belonging to the associated learner archive).

The multi-individual (team-based) solutions of the CMGE models evolved in this

thesis were analyzed in terms of their inter and intra-class behaviors. We first noted that the CMGE inter-class behavior differed substantially from that of the PGEC models in the discussion at the end of Chapter 5 regarding the PGEC classification results supplied in Appendix G. In compiling these results the PGEC teams were observed to be extremely inconsistent between classes and initializations in terms of performance and team sizes. This highlighted substantial differences with regards to team compositions and voting policies. We noted that the CMGE learners (of all classes) evolve together, seeing all of the relevant point archives at once while learning strong voting behaviors leading to a winner-take-all voting policy. In comparison, PGEC learners can only use one point archive per class-specific initialization where there are no guarantees on the relevance of out-of-class points. Moreover, its variable sized teams of weak learners tended to negatively influence the majority voting procedure. The end result was very poor classification performance in comparison to our CMGE models.

Intra-class voting behaviors were analyzed in Chapter 8 where evidence for 'problem decomposition' voting behaviors of CMGE were contrasted with the 'weak learning' of PGEC. A key difference with respect to intra-class behaviors is that CMGE models are encouraged to learn strong, class consistent voting in accordance with its winner-take-all voting policy, where no such requirement is enforced by PGEC. This behavior is attributed to CMGE's use of real-valued outcomes (based on LMF output) as entry criteria to learner archives. Moreover, the cooperative (coverage) behavior of CMGE is achieved through the explicit minimization of overlap objective employed in the EMO component. This combination of good inter and intra-class voting behavior allows our model to address the problem decomposition issue without sacrificing classification performance.

10.2.5 Multi-class Applications

The conventional approach to multi-class problems under the canonical GP classification context involves execution of multiple independent initializations so that each class is handled as a separately evolved binary classifier, thus requiring N runs of GP for an N-class problem [58]. Aside from the added computational overhead associated

with N independent runs (indicated in Equation 1.2 of Chapter 1), such an approach also assumes an appropriate combination (output) policy can be specified such that a single response (class label) is provided for each input vector. This issue has been demonstrated in the current thesis where we choose to adopt either a sigmoid wrapper approach combined with a winner-take-all voting policy (in the case of baseline models introduced in Chapter 4) or a majority voting collection of separately evolved, class-specific teams (in the case of the binary PGEC model). In either case, training required the added steps of multiple initializations and re-labelling the training data to generate solutions for each class, resulting in systems with unnecessary complexity and computational overhead.

The multi-class objective was addressed by the multi-class archiving and a 'novelty detection' approach taken by the current framework which permits allocation of individuals to any class according to results of clustering the raw GP output, as described in Chapter 3. In other words, evolution of multi-class individuals takes place in parallel with multi-class learner archiving leading to a single training run providing solutions for all classes. Moreover, class-specific stopping criteria are used to ensure sufficient training in each class and efficient allocation of population resources. Deployment of the resulting classifiers also takes place in parallel, employing a winner-take-all policy where this requires stronger performance of the classifiers as a whole than the hierarchical case, which can mask poor performance of classifiers appearing later in the hierarchy. A principal benefit of the parallel classification model is that it provides us with the inherent ability to address the multi-label classification domains in which exemplars might be a member of multiple classes.

Evidence of the appropriateness of our multi-class approach has been discussed above, with strong training and test classification performance returned in the results of Chapter 5 for both binary and multi-class problems as compared to the similarly parameterized and constrained baseline and PGEC models. Moreover, the class-wise detection and false positive rates provided in Appendix G readily support the strong (overall) score and accuracy based performances.

10.3 Contributions

The high-level contribution of this thesis is clearly the development of a classification framework for GP that addresses our five research objectives outlined in Chapter 1. Within this framework, however, we additionally consider several algorithmic design details to be significant in terms of providing novel contributions to the GP community:

Novelty detection : We have established the suitability and generally appealing properties of a local (Gaussian) membership function approach to wrapper function design for classification. In the current work the parameters defining the LMF are driven by clustering the raw (one dimensional) GP outputs, however any similar approach for estimating the GP's output distribution could be equally applicable;

Explicit problem decomposition : A multi-objective approach that specifically encourages cooperative problem decomposition through an explicit 'overlap' objective has been established. This objective might be redefined to make better use of archive information in future work (Section 10.5);

Competitive multi-class training : The extension of Lemczyk's binary PGPC [70] to a parallel, multi-class Pareto coevolutionary approach that incorporates real-valued outcomes driving class-specific archiving has been demonstrated to provide strong voting behaviors while establishing the basis for sub sampling of the data, lending to scalability. The current work employs a balanced sampling heuristic but this could be improved in future work (Section 10.5);

Early stopping with reallocation of resources : Class-specific evaluation of early stopping criteria (based on the Kumar and Rockett histogram technique [66]) allows the dynamic redistribution of population members by way of re-initializations during evolution. This design has obvious implications for efficiency and can allow for focusing of resources on more difficult classes during training.

10.4 Classification with GP

In light of the No Free Lunch theorems (Section 1.2.3), it is important to consider under what circumstances GP might be an appropriate choice for the classification task. In general we have made the case for the flexibility of GP in terms of its allowances for user-defined representation and cost function. Moreover, the highly explorative approach to the credit assignment associated with stochastic search along with its capacity for resilience to convergence on local optima and few requirements on training data have been highlighted as potential strengths. With the current framework for classification addressing the classical pathologies (specifically the algorithmic complexity typically associated with stochastic search) we can now make a case for the GP approach when these potentially advantageous features are desired in the training algorithm.

We have seen in the results of this thesis that often a simple model (e.g., the Linear Perceptron trained with a robust second order algorithm, or the C4.5 decision tree classifier) can provide the best classification results while being extremely reliable and efficient in terms of computational overhead; however, the CMGE models consistently returned among the best results without pre-processing of data, specification of solution architecture, tuning of algorithm parameters or assumptions in terms of data distributions. Moreover, with the computational overhead being reduced to levels that are comparable to other ML classifiers, one of the primary considerations perhaps becomes constraints and representational biases due to the solution form. Here the current framework makes use of the GE formulation of GP, which provides a convenient grammar-based (CFG) definition of the output language that is able to elegantly addresses the syntactic closure requirement. In addition to this high degree of representational flexibility, the CMGE framework provides automatic multi-class problem decomposition behavior as opposed to monolithic (single model) solutions, while placing no specific constraints on the feature usage. This special modular problem decomposition behavior may have particular advantages where sub problems exist within the classification data or when post-training analysis or specific deployment conditions can benefit from multiple expressions cooperating to solve the problem.

Given the algorithmic advantages introduced to the GP classification algorithm

through the CMGE framework, an argument can be made that in a situation where very little *a priori* knowledge exists to prefer any of the specific ML classifier over the others, the GP framework presented in this thesis may at minimum provide a strong starting point for model induction.

10.5 Recommendations for Future Work

Finally, it is intended that this framework provide continued motivation and directions for further research into the classification context of Genetic Programming. Expanding on the points outlined in Section 10.3, immediate recommendations for future work might include investigation into algorithms for efficiently guiding the sampling of the exemplars beyond the default 'balanced' sampling heuristic employed here. Of specific interest might be the potential to efficiently maintain difficulty information that is based on, for example, the misclassification rates of points as they are sampled from the training data. The benefits of such a mechanism would be two-fold. First, the difficulty information that could be gained during the early (largely undirected) sampling of points could be used (e.g., stochastically) to guide the search toward 'useful' exemplars later in evolution, potentially accelerating training when the requirement for more difficult points would normally leave a more naïve sampling heuristic searching blindly for points to maintain a training gradient. Secondly, the difficulty information related to points, post training, could lend further insight into the nature of the learning process and (perhaps more importantly) the classification problem itself.

An additional straightforward recommendation might be that the 'overlap' objective be improved to incorporate the information of the real-valued outcomes from the learner archive to arrive at equally efficient statistics that could be developed to provide a more accurate picture of the archive coverage. This could have the potential to effect the cooperative problem decomposition behavior in much the same way that the use of real-valued outcomes as the basis for archive entry focused or strengthened the voting behavior in the current work.

In terms of more specialized algorithm modifications, the clustering parameters of the Potential Function (specifically α and β) might be implemented so as to allow

for their automatic adjustment according to basic cluster validation metrics, for example inter-cluster separation distance and/or intra-cluster variation. Moreover, the clusterings returned by the Potential Function might be analyzed for their potential to detect points from multiple classes, thus reducing the numbers of individuals participating in team solutions while further reducing the computational overhead when classifiers are deployed, post-training.

As always, the author hopes to inspire further investigation into this line of research, including alternate configurations and novel applications of the current framework, and endeavors to pursue the potential of Genetic Programming to support a wide range of real-world tasks through flexible and efficient evolutionary modeling.

Bibliography

[1] Shawkat Ali and Kate A. Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6:119–138, 2006.

[2] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, October 2004.

[3] Peter J. Angeline and Jordan B. Pollack. Competitive environments evolve better solutions for complex tasks. *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 264–270, 1993.

[4] Jim Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 86–91, 1989.

[5] T. Back, U. Hammel, and H. P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1:3–17, 1997.

[6] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming - An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann, 1997.

[7] F. H. Bennett, John R. Koza, J. Shipman, and O. Stiffelman. Building a parallel computer system for $18,000 that performs a half peta-flop per day. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-1999*, pages 1484–1490, 1999.

[8] Mark L. Berenson, M. Goldstein, and D. Levine. *Intermediate Statistical Methods and Applications: A Computer Package Approach*. Prentice Hall College Div, December 1982.

[9] Ester Bernado-Mansilla and Josep M. Garrell-Guiu. Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks. *Evolutionary Computation*, 11:209–239, 2003.

[10] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1996.

[11] Stefan Bleuler, Martin Brack, Lothar Thiele, and Eckart Zitzler. Multiobjective genetic programming: Reducing bloat using SPEA2. In *CEC 2001*, pages 536–543, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 2001. IEEE Press.

[12] Celia C. Bojarczuk, Heitor S. Lopes, and Alex A. Freitas. An innovative application of a constrained syntax genetic programming system to the problem of predicting survival of patients. *EuroGP 2003*, pages 11–21, 2003.

[13] Celia C. Bojarczuk, Heitor S. Lopes, Alex A. Freitas, and Edson L. Michalkiewicz. A constrained-syntax genetic programming system for discovering classification rules: Application to medical data sets. *Artificial Intelligence in Medicine*, 30:27–48, 2004.

[14] J. C. Bongard and H. Lipson. Managed challenge alleviates disengagement in coevolutionary system identification. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-05*, 1:531–538, 2005.

[15] M. Brameier and W. Banzhaf. A comparison of linear genetic programming and neural networks in medical data mining. *IEEE Transactions on Evolutionary Computation*, 5:17–26, 2001.

[16] M. Brameier and W. Banzhaf. Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2:381–407, 2001.

[17] John Cartlidge and Seth Bullock. Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. *IEEE Congress on Evolutionary Computation*, 2:1420–1425, 2002.

[18] Stephen L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2:267–278, 1994.

[19] Steffen Christensen and Mark Wineberg. Using appropriate statistics - statistics for artificial intelligence. In *Tutorial Program of 2004 Genetic and Evolutionary Computation Conference*, pages 544–564, Seattle, WA., 2004.

[20] Carlos A. Coello Coello. Evolutionary muli-objective optimization; a historical view of the field. *IEEE Computational Intelligence*, 1:28–36, 2006.

[21] R. Curry and M. I. Heywood. Towards efficient training on large datasets for genetic programming. *17th Conference of the Canadian Society for Computational Studies of Intelligence, LNAI*, 3060:161–174, 2004.

[22] R. Curry, P. Lichodzijewski, and M. I. Heywood. Scaling genetic programming to large datasets using hierarchical dynamic subset selection. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 37:1065–1073, 2007.

[23] Edwin de Jong. The incremental pareto-coevolution archive. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-04*, pages 525–536, 2004.

[24] Edwin de Jong and Jordan Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, 12:159–192, 2004.

[25] Edwin de Jong, Richard A. Watson, and Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 11–18, 2001.

[26] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, February 2001.

[27] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGAII. *IEEE Transactions on Evolutionary Computation*, 6:182–197, April 2002.

[28] Howard Demuth and Mark Beale. *Neural Network Toolbox For Use with MATLAB, User's Guide (Version 4)*. The MathWorks, 2001.

[29] Chris Drummond. Machine learning as an experimental science. *Evaluation Methods for Machine Learning (AAAI Workshop)*, pages 1–6, 2006.

[30] C. Elkan. Results of the KDD99 classifier learning contest.

[31] C. Elkan. Boosting and naive bayesian learning. *Technical Report No. CS97-557, UCSD*, 1997.

[32] I. De Falco, A. Della Cioppa, and E. Tarantino. Discovering interesting classification rules with genetic programming. *Applied Soft Computing*, 1:257–269, 2001.

[33] Francisco Fernandez, Leonardo Vanneschi, and Marco Tomassini. The effect of plagues in genetic programming: A study of variable-size populations. *EuroGP 2003*, pages 317–326, 2003.

[34] Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. *Sixth European Conference on Artificial Life*, pages 316–325, 2001.

[35] Sevan G. Ficici and Jordan B. Pollack. A game-theoretic memory mechanism for coevolution. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-03*, pages 286–297, 2003.

[36] Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano. Improving cooperative GP ensemble with clustering and pruning for pattern classification. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:791–798, 2006.

[37] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423, 1993.

[38] Alex A. Freitas. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, October 2002.

[39] R. M. Friedberg. A learning machine: Part I. *IBM Journal of Research and Development*, 2:2–13, 1958.

[40] Christian Gagne, Marc Schoenauer, Michele Sebag, and Marco Tomassini. Genetic programming for kernel-based learning with co-evolving subsets selection. *Parallel Problem Solving from Nature IX*, pages 1008–1017, 2006.

[41] Christian Gagne, Marc Schoenauer, and Marco Tomassini. Ensemble learning for free with evolutionary algoirthms? *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-07)*, pages 1782–1789, 2007.

[42] Chris Gathercole and Peter Ross. Dynamic training subset selection for supervised learning in genetic programming. *Parallel Problem Solving from Nature III*, 866:312–321, 1994.

[43] Ashley George and M. I. Heywood. Improving GP classifier generalization using a cluster separation metric. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:939–940, 2006.

[44] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[45] Jiawei Han and Micheline Kamber. *Data Mining : Concepts and Techniques*. Morgan Kaufmann, second edition, March 2006.

[46] L.K. Hansen and P. Salamon. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12:993–1001, 1990.

[47] Robin Harper and Alan Blair. A structure preserving crossover in grammatical evolution. *IEEE Congress on Evolutionary Computation*, 1:2537–2544, 2005.

[48] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, July 1998.

[49] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, fourth edition, September 2006.

[50] S Hettich and S. D. Bay. The UCI KDD archive, 1999.

[51] M.I. Heywood and A.N. Zincir-Heywood. Dynamic page based crossover in linear genetic programming. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 32:380–388, 2002.

[52] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–91, 1993.

[53] A. W. Iorio and X. Li. A cooperative coevolutionary multiobjective algorithm using non-dominated sorting. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-04*, 2:537–548, 2004.

[54] Nathalie Japkowicz. Why question machine learning evaluation methods. *Evaluation Methods for Machine Learning (AAAI Workshop)*, pages 6–11, 2006.

[55] Yaochu Jin. *Multi-Objective Machine Learning (Studies in Computational Intelligence)*. Springer, March 2006.

[56] Hugues Juille and Jordan B. Pollack. Coevolving the ideal trainer: Application to the discovery of cellular automata rules. *Proceedings of the Third Annual Conference on Genetic Programming*, pages 519–527, 1998.

[57] H. Gunes Kayacik, A. Nur Zincir-Heywood, and Malcolm I. Heywood. A hierarchical som-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, 20:439–451, 2007.

[58] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation*, 4:242–248, 2000.

[59] M. Kotanchek, G. Smits, and E. Vladislavleva. *Pursuing the Pareto Paradigm: Tournaments, Algorithm Variations and Ordinal Optimization*, pages 167–185. Genetic and Evolutinary Computation. 0387333754, March 2007.

[60] John R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, 1:768–774, 1989.

[61] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. The MIT Press, December 1992.

[62] John R. Koza. *Genetic Programming II : Automatic Discovery of Reusable Programs*. The MIT Press, May 1994.

[63] John R. Koza, Forrest H Bennett III, Jeffrey L. Hutchings, Stephen L. Bade, Martin A. Keane, and David Andre. Evolving computer programs using rapidly reconfigurable fpgas and genetic programming. In Jason Cong, editor, *Sixth International Symposium on Field Programmable Gate Arrays*, pages 209–219, Doubletree Hotel, Monterey, California, USA, 1998. ACM Press.

[64] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, 7:231–238, 1995.

[65] Rajeev Kumar and Peter Rockett. Multiobjective genetic algorithm partitioning for hierarchical learning of high-dimensional pattern spaces: A learning-follows-decomposition strategy. *IEEE Transactions on Neural Networks*, 9:822–830, September 1998.

[66] Rajeev Kumar and Peter Rockett. Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: A pareto converging genetic algorithm. *Evolutionary computation*, 10:283–314, 2002.

[67] William B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer, March 2002.

[68] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10:263–282, 2002.

[69] Y. LeCun, L. Bottou, G. B. Orr, and K.R. Mueller. Efficient backprop. *Lecture Notes in Computer Science*, 1524:9–50, 1998.

[70] Michal Lemczyk and M. I. Heywood. Pareto-coevolutionary genetic programming classifier. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:945–946, 2006.

[71] Michal Lemczyk and M. I. Heywood. Training binary GP classifiers efficiently: a pareto-coevolutionary approach. *Proceedings of the 10th European Conference on Genetic Programming, EuroGP-07*, pages 229–240, 2007.

[72] P. Lichodzijewski and M. I. Heywood. GP classifier problem decomposition using first-price and second-price auctions. *Proceedings of the 10th European Conference on Genetic Programming, EuroGP-07*, pages 137–147, 2007.

[73] P. Lichodzijewski and M. I. Heywood. Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-07*, 1:464–471, 2007.

[74] Tjen-Sien Lim, Wei-Yin Loh, and Yu-Shan Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 40:203–228, 2000.

[75] Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. *Proceedings of the Congress on Evolutionary Computation 2001*, 2:1070–1077, 2001.

[76] Sean Luke and Gabriel Catalin Balan. Population implosion in genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-03*, pages 1729–1739, 2003.

[77] Sean Luke and Liviu Panait. Lexicographic parsimony pressure. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-02*, pages 829–836, 2002.

[78] Prechelt Lutz. Proben 1 - a set of benchmarks and benchmarking rules for neural network training algorithms. *Univ. Karlsruhe (Karlsruhe, Germany)*, 1994.

[79] Markos Markou and Sameer Singh. Novelty detection: a review - part 2: neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[80] A. R. McIntyre and M. I. Heywood. On multi-class classification by way of niching. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-04*, 2:581–592, 2004.

[81] A. R. McIntyre and M. I. Heywood. Toward co-evolutionary training of a multi-class classifier. *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pages 2130–2137, 2005.

[82] A. R. McIntyre and M. I. Heywood. Moge: GP classification problem decomposition using multi-objective optimization. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-06*, 1:863–870, 2006.

[83] William Mendenhall and Terry Sincich. *Statistics for the Engineering and Computer Sciences*. Macmillan Pub Co, second edition, April 1988.

[84] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, second edition, September 1994.

[85] Brad L. Miller and Michael J. Shaw. Genetic algorithms with dynamic niche sharing for multimodal function optimization. *International Conference on Evolutionary Computation*, pages 786–791, 1996.

[86] Melanie Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, February 1998.

[87] D.P. Muni, N.R. Pal, and J. Das. A novel approach to design classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation*, 8:183–196, 2004.

[88] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI machine learning repository.

[89] Jason Noble and Richard Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-01*, pages 493–500, 2001.

[90] Peter Nordin. *A Compiling Genetic Programming System that Directly Manipulates the Machine Code*, pages 311–332. MIT Press, Cambridge, MA, 1994.

[91] Michael O'Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Springer, May 2003.

[92] David W. Opitz and Jude W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. *Advances in Neural Information Processing Systems*, 8:535–541, 1996.

[93] Daniel Parrott, Li Xiaodong, and Victor Ciesielski. Multiobjective techniques in genetic programming for evolving classifiers. *IEEE Congress on Evolutionary Computation*, 2:1141–1148, 2005.

[94] T.K. Paul, Y. Hasegawa, and H. Iba. Classification of gene expression data by majority voting genetic programming classifier. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 2521– 2528, 2006.

[95] Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8:1–29, 2000.

[96] Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[97] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 616–623, 2003.

[98] Craig W. Reynolds. Competition, coevolution and the game of tag. *Proceedings of Artificial Life IV*, pages 59–69, 1994.

[99] K Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming. Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. *IEEE Transactions on Systems, Man, and Cybernetics*, 34:531–545, July 2004.

[100] Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5:1–29, 1997.

[101] Gunter Rudolph. Convergence analysis of canonical genetic algorithms. *Neural Networks, IEEE Transactions on*, 5:96–101, 1994.

[102] Suseela T. Sarasamma, Qiuming A. Zhu, and Julie Huff. Hierarchical kohonen net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics*, 35:302–312, April 2005.

[103] David Schaffer. *Multiple objective optimization with Vector Evaluated Genetic Algorithms*. Ph.d, Vanderbilt University, 1984.

[104] David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. *Genetic Algorithms and the Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[105] William Smart and Mengjie Zhang. Using genetic programming for multiclass classification by simultaneously solving component binary classification problems. *EuroGP 2005*, pages 227–239, 2005.

[106] Matthew G. Smith and Larry Bull. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines*, 6:265–281, 2005.

[107] G. Smits and E. Vladislavleva. Ordinal pareto genetic programming. *IEEE Congress on Evolutionary Computation (CEC)*, pages 3114–3120, 2006.

[108] D. Song, M. I. Heywood, and A. N. Zincir-Heywood. Training genetic programming on half a million patterns: An example from anomaly detection. *IEEE Transactions on Evolutionary Computation*, 9:225–239, 2005.

[109] Terence Soule. Heterogeneity and specialization in evolving teams. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2000*, pages 778–785, 2000.

[110] Terence Soule. Cooperative evolution on the intertwined spirals problem. *Genetic Programming: 6th European Conference Proceedings*, 2610:434–442, 2003.

[111] Terence Soule and Pavankumarreddy Komireddy. *Orthogonal Evolution of Teeams: A Class of Algorithms for Evolving Teams with Inverseley Correlated Errors*, pages 79–95. Genetic and Evolutionary Computation. Springer, March 2007.

[112] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248, 1995.

[113] Gilbert Syswerda. *A Study of Reproduction in Generational and Steady State Genetic Algorithms*, volume 1, pages 94–101. Morgan Kaufmann, July 1991.

[114] T.Z. Tan, C. Quek, and G.S. Ng. Brain-inspired genetic complementary learning for stock market prediction. *IEEE Congress on Evolutionary Computation, 2005.*, 3:2653– 2660, 2005.

[115] Richard Watson and Jordan Pollack. Coevolutionary dynamics in a minimal substrate. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-01*, pages 702–709, 2001.

[116] Gary M. Weiss and Foster Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–345, 2003.

[117] Leigh Wetmore, M. I. Heywood, and A. N. Zincir-Heywood. Speeding up the self-organizing feature map using dynamic subset selection. *Neural Processing Letters*, 22:17–22, 2005.

[118] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, second edition, June 2005.

[119] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. on Evolutionary Computation*, 1:67–82, 1997.

[120] Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence: Advanced Topics in Artificial Intelligence*, pages 180–192, 1999.

[121] Yang Zhang and Peter Rockett. Feature extraction using multi-objective genetic programming. *Studeies in Computational Intelligence*, 16:79–106, 2006.

[122] Zijian Zheng. A benchmark for classifier learning. *Technical Report TR474 (N.S.W Australia 2006)*, 1993.

[123] E. Zitzler, M. Laumanns, and S. Bleuler. *A Tutorial on Evolutionary Multiobjective Optimization*, pages 3–38. Springer-Verlag, Berlin, Germany, 2004.

[124] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. In *EUROGEN 2001.*, pages 95–100, Athens, Greece, 2002.

[125] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms - a comparative case study. In *PPSN V*, volume 1498, pages 292–301, Amsterdam, 1998. Springer-Verlag.

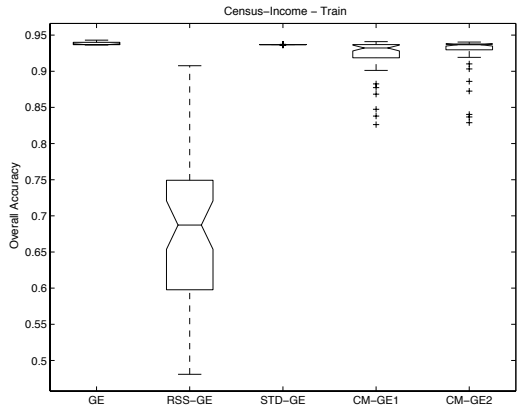
[126] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3:257–271, 1999.

Appendix A

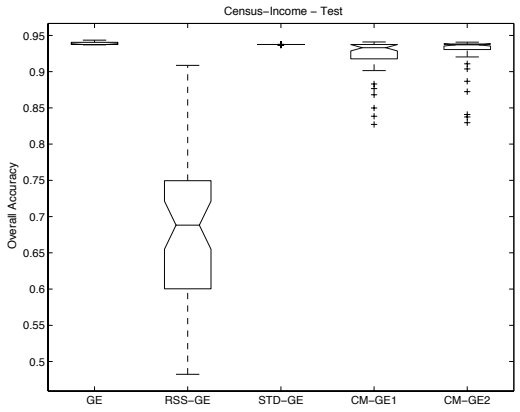
GP Comparison Plots (E1 - E5)

A.1 Canonical GP Comparisons

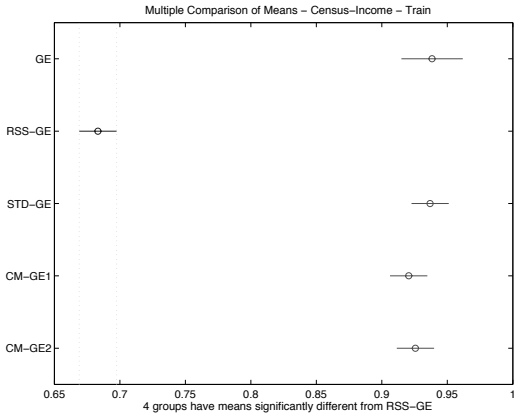
A.1.1 Overall Accuracy



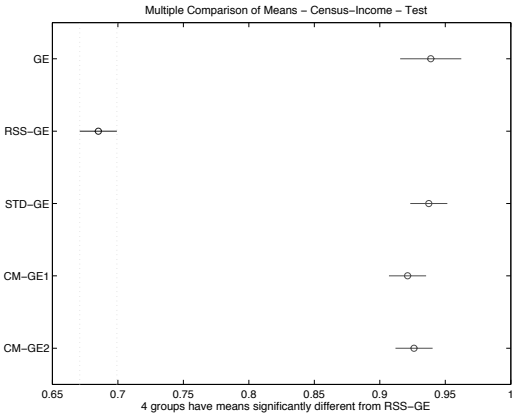
(a) Overall Accuracy (Train)



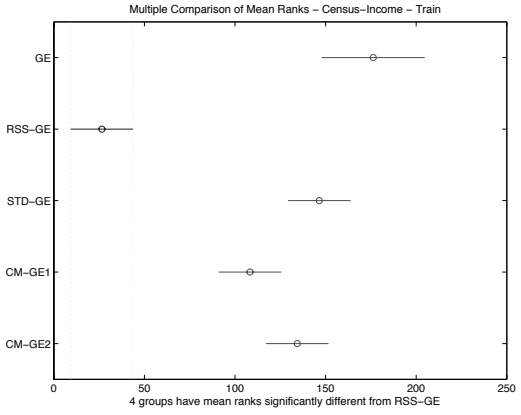
(b) Overall Accuracy (Test)



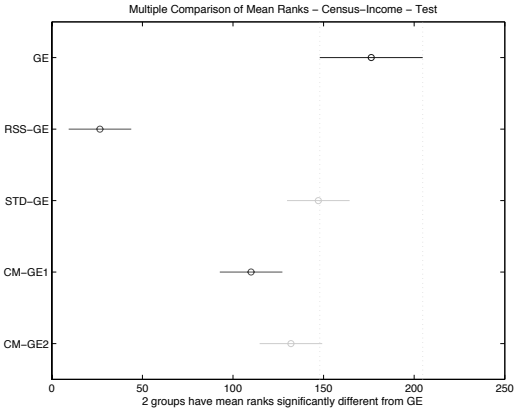
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

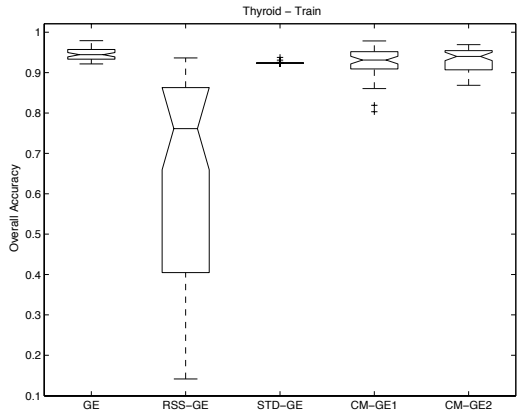


(e) Comparison of Mean Ranks (Train)

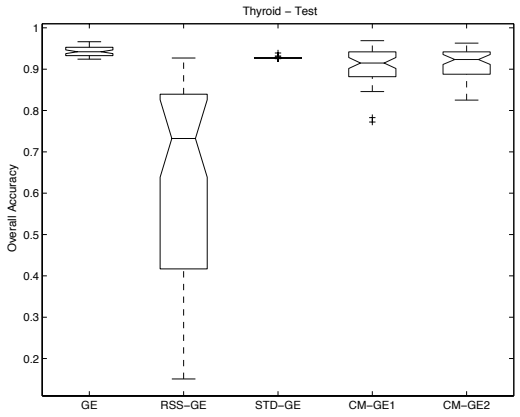


(f) Comparison of Mean Ranks (Test)

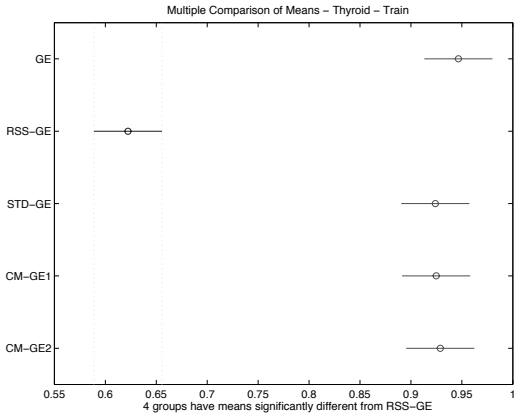
Figure A.1: Direct (GE) comparison of CENS Overall Accuracy performance.



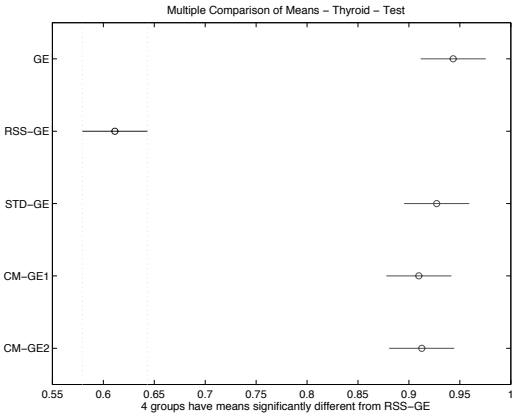
(a) Overall Accuracy (Train)



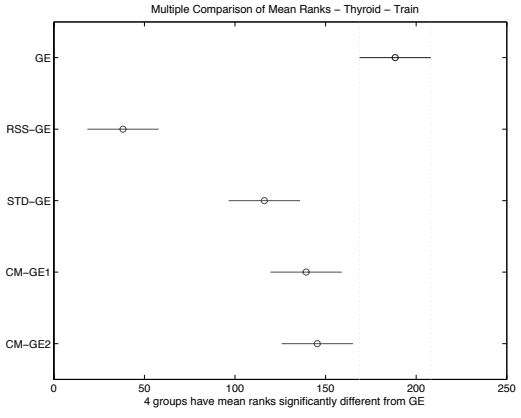
(b) Overall Accuracy (Test)



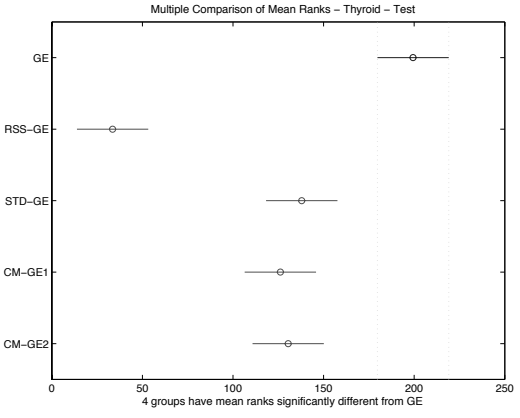
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



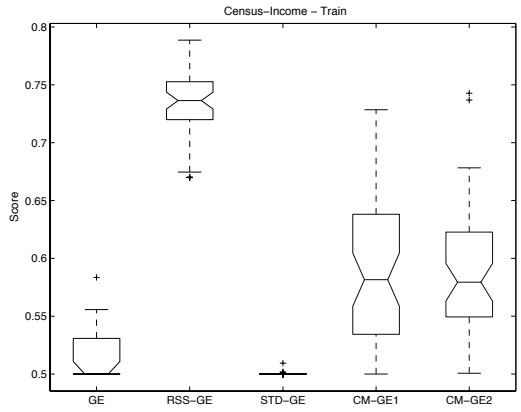
(e) Comparison of Mean Ranks (Train)



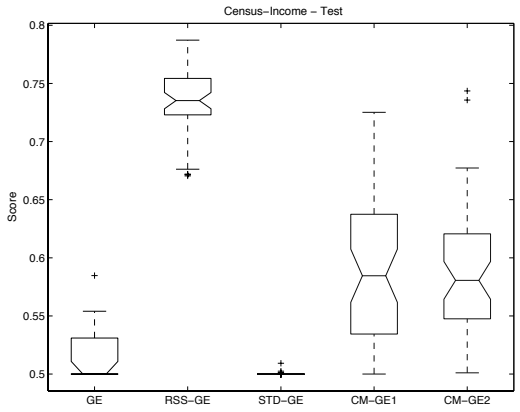
(f) Comparison of Mean Ranks (Test)

Figure A.2: Direct (GE) comparison of THYD Overall Accuracy performance.

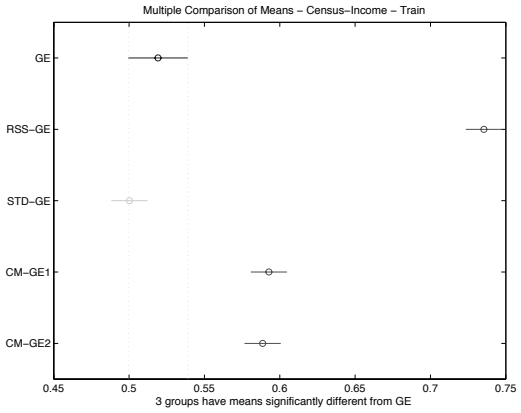
A.1.2 Score



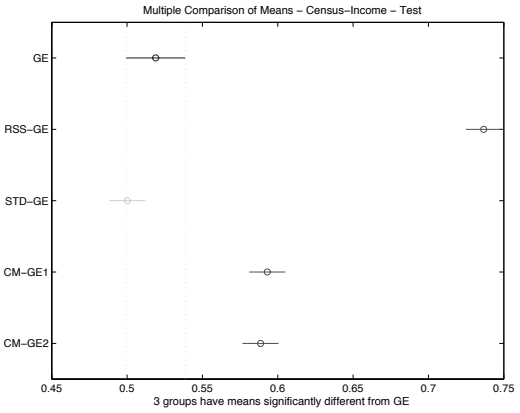
(a) Score (Train)



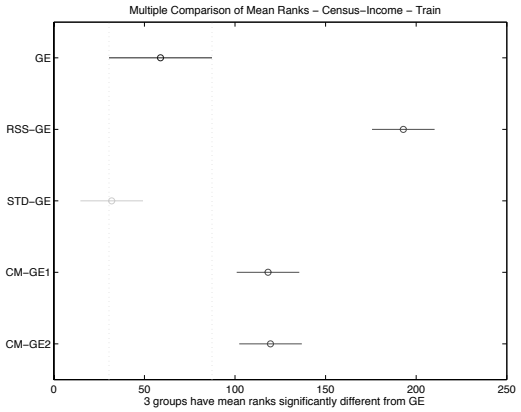
(b) Score (Test)



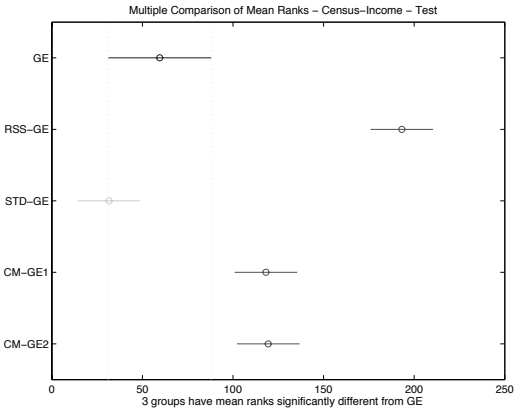
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

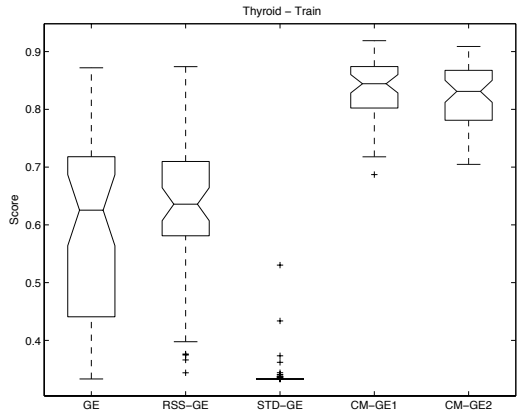


(e) Comparison of Mean Ranks (Train)

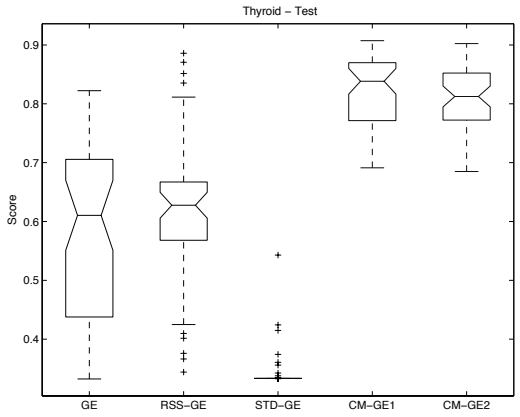


(f) Comparison of Mean Ranks (Test)

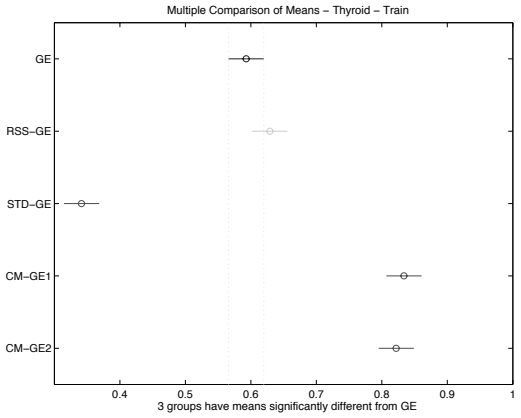
Figure A.3: Direct (GE) comparison of CENS Score performance.



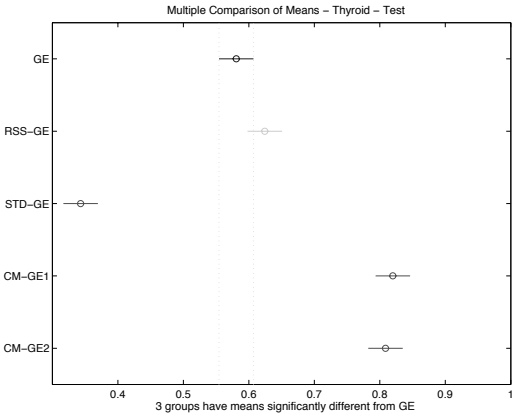
(a) Score (Train)



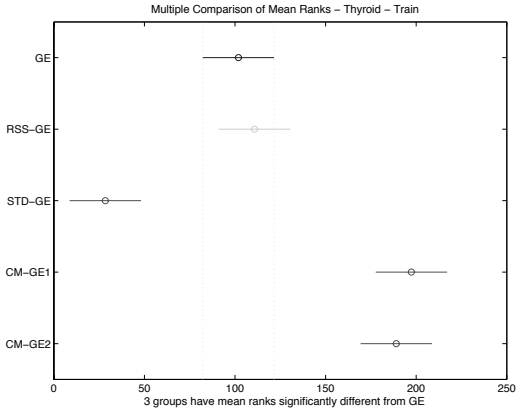
(b) Score (Test)



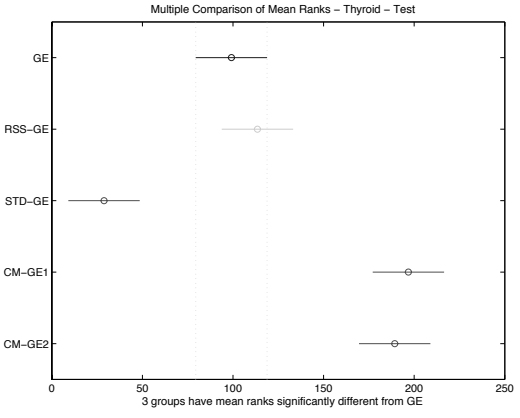
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



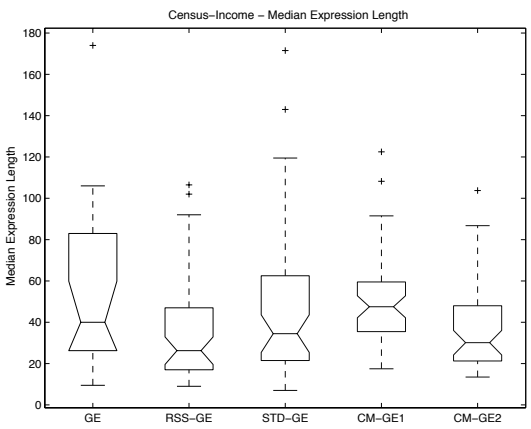
(e) Comparison of Mean Ranks (Train)



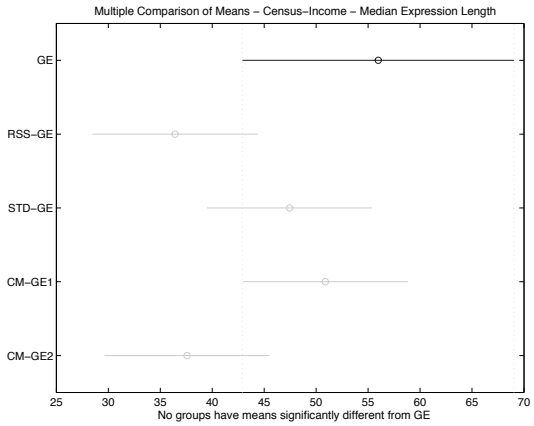
(f) Comparison of Mean Ranks (Test)

Figure A.4: Direct (GE) comparison of THYD Score performance.

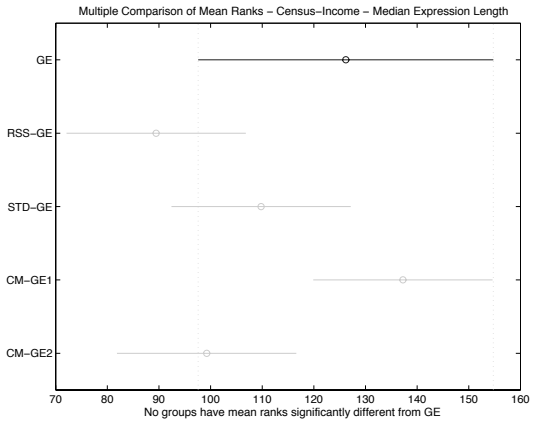
A.1.3 Solution Complexity (String length)



(a) Solution Length (strlen)

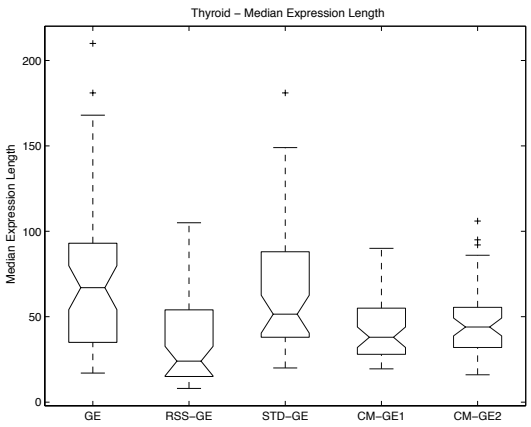


(b) Comparison of Means

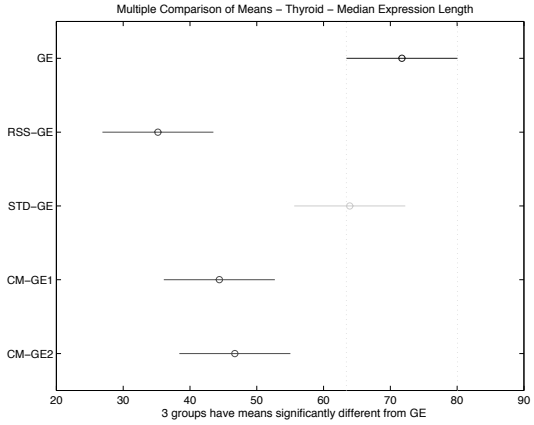


(c) Comparison of Mean Ranks

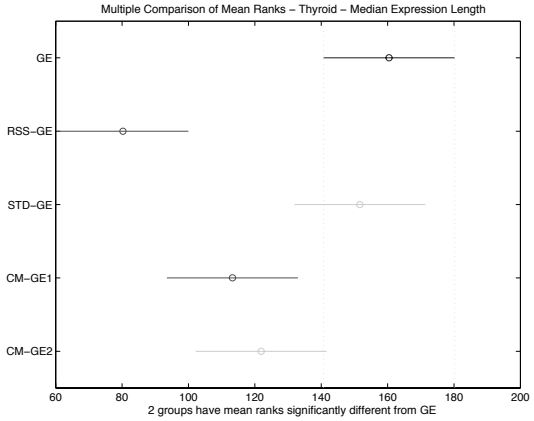
Figure A.5: Direct (GE) comparison of solution length on CENS.



(a) Solution Length (strlen)



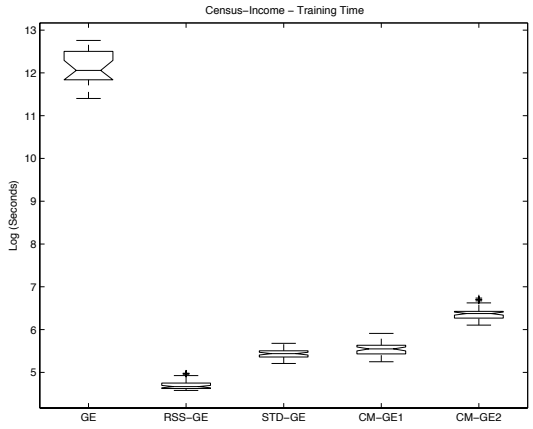
(b) Comparison of Means



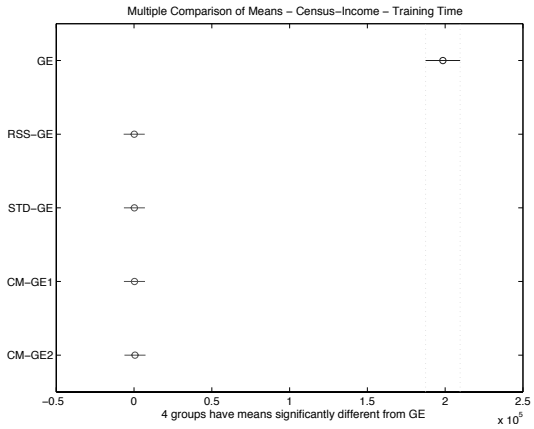
(c) Comparison of Mean Ranks

Figure A.6: Direct (GE) comparison of solution length on THYD.

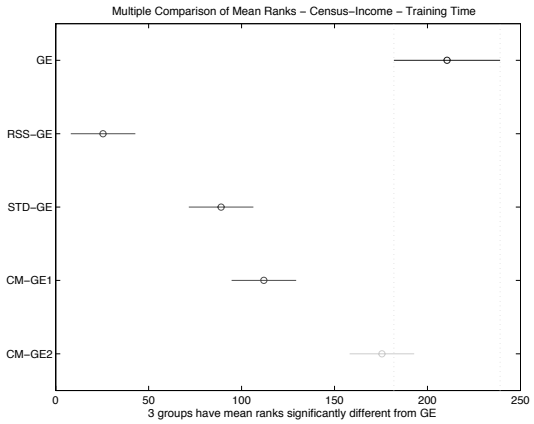
A.1.4 Training Time



(a) Training Time

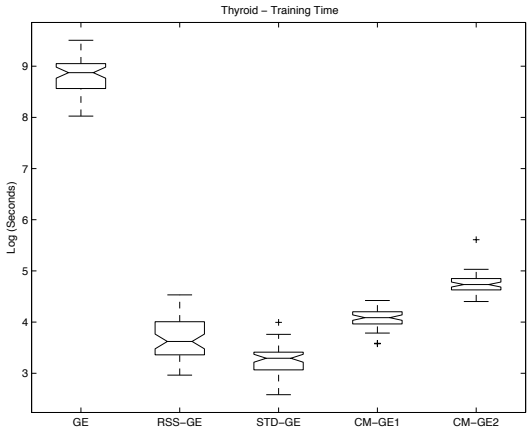


(b) Comparison of Means

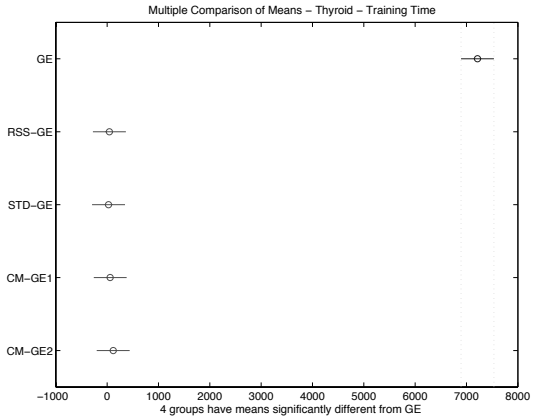


(c) Comparison of Mean Ranks

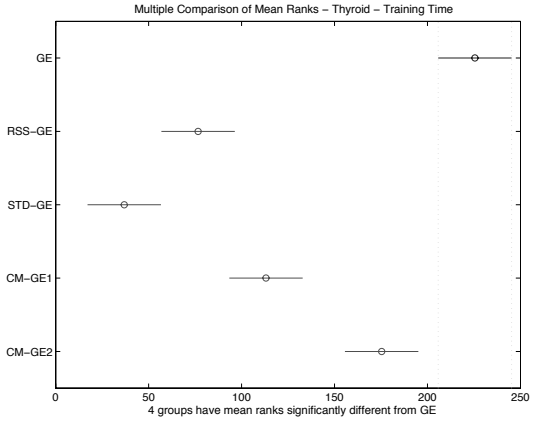
Figure A.7: Direct (GE) comparison of training time on CENS.



(a) Training Time



(b) Comparison of Means

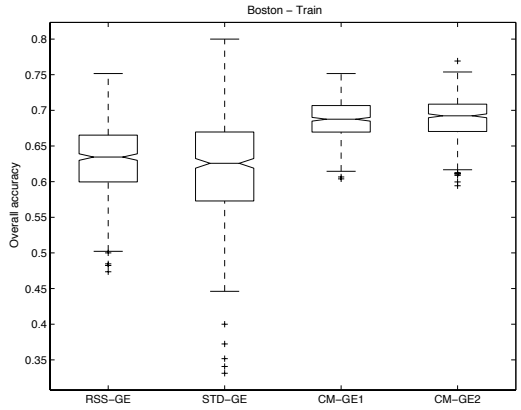


(c) Comparison of Mean Ranks

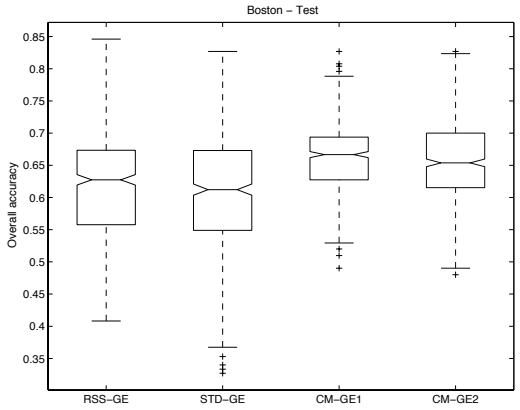
Figure A.8: Direct (GE) comparison of training time on THYD.

A.2 Scalable GP Comparisons

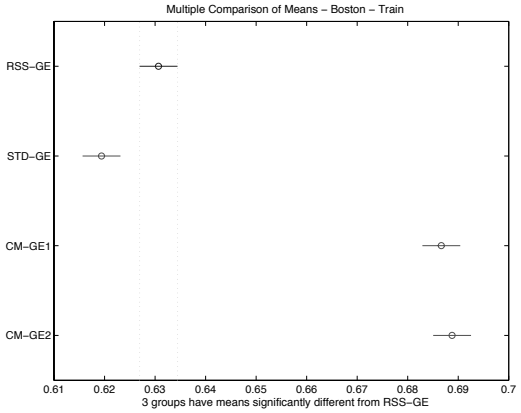
A.2.1 Overall Accuracy



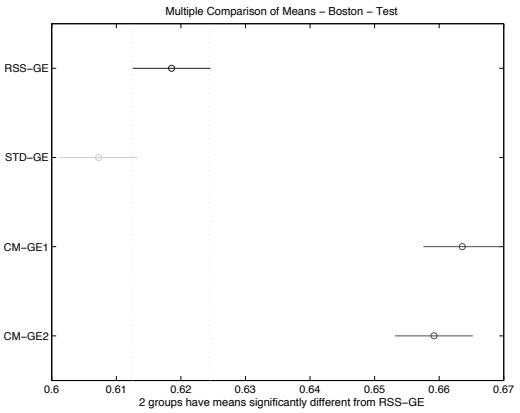
(a) Overall Accuracy (Train)



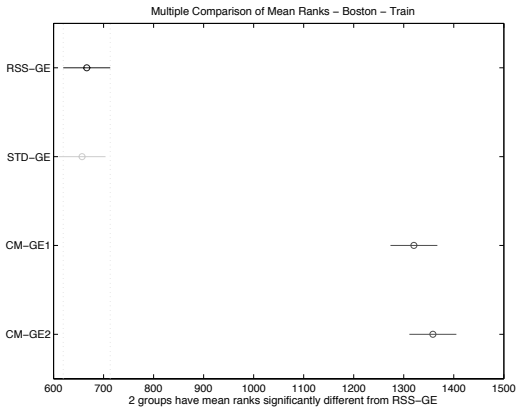
(b) Overall Accuracy (Test)



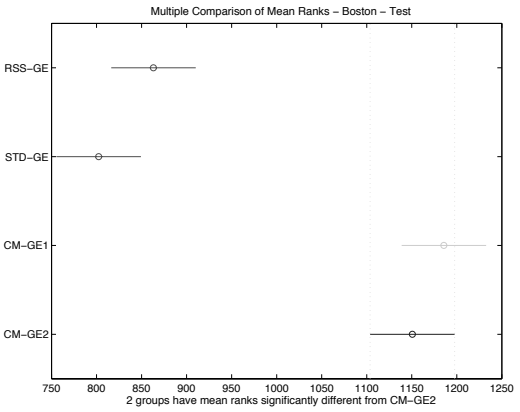
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

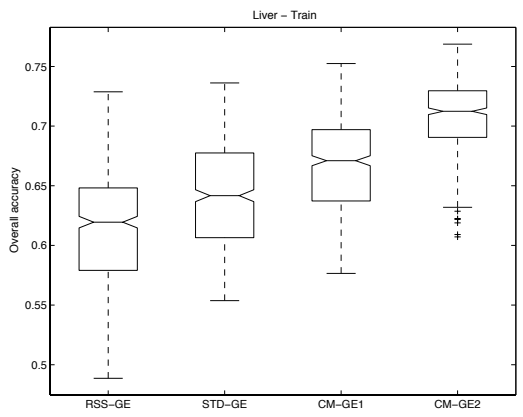


(e) Comparison of Mean Ranks (Train)

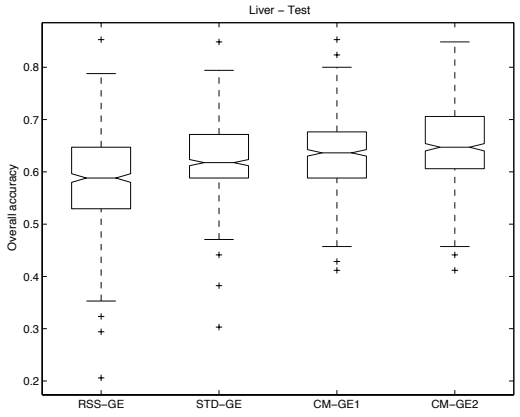


(f) Comparison of Mean Ranks (Test)

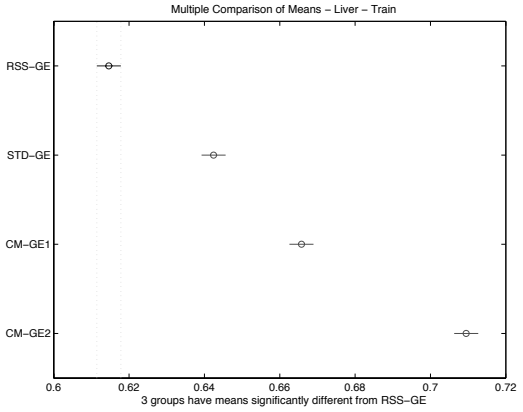
Figure A.9: Direct (GE) comparison of BOST Overall Accuracy performance.



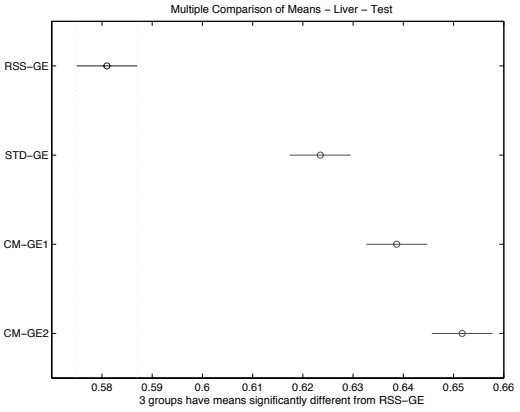
(a) Overall Accuracy (Train)



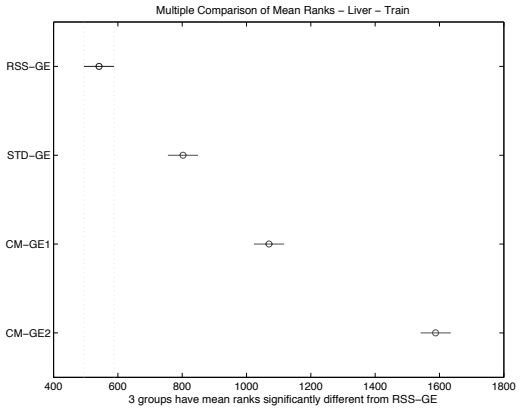
(b) Overall Accuracy (Test)



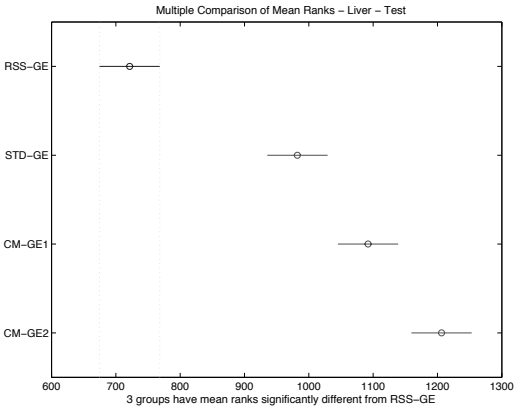
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

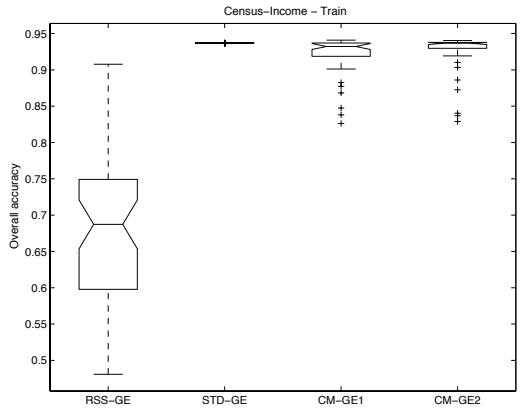


(e) Comparison of Mean Ranks (Train)

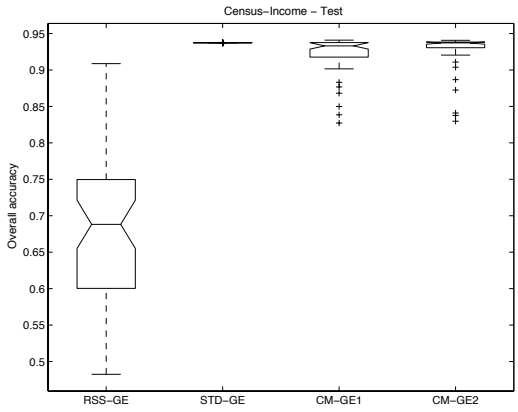


(f) Comparison of Mean Ranks (Test)

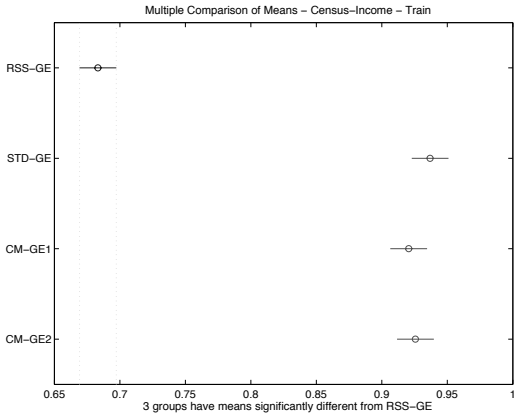
Figure A.10: Direct (GE) comparison of BUPA Overall Accuracy performance.



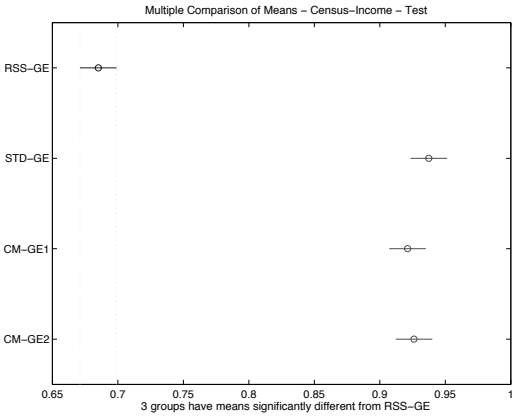
(a) Overall Accuracy (Train)



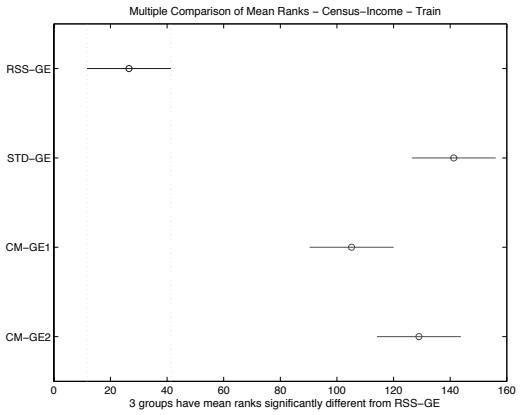
(b) Overall Accuracy (Test)



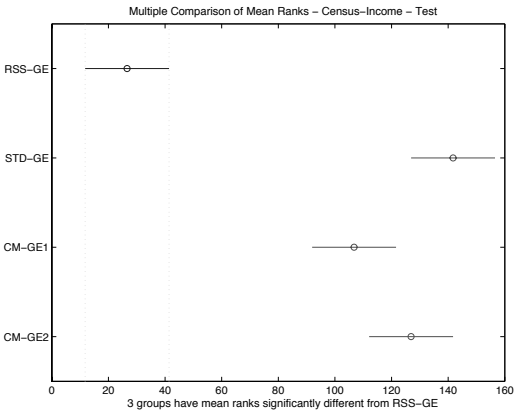
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

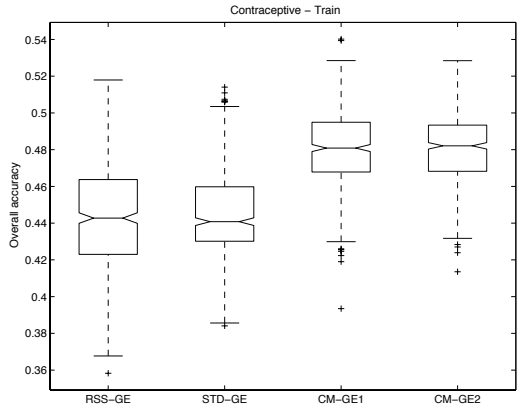


(e) Comparison of Mean Ranks (Train)

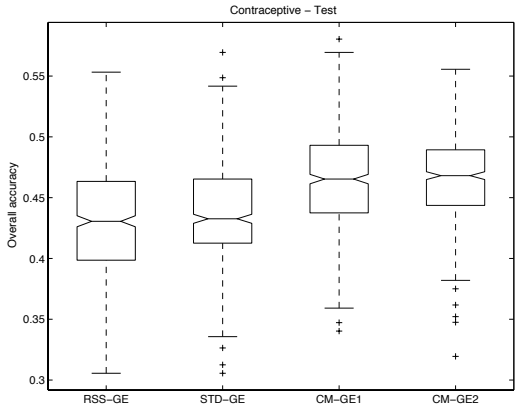


(f) Comparison of Mean Ranks (Test)

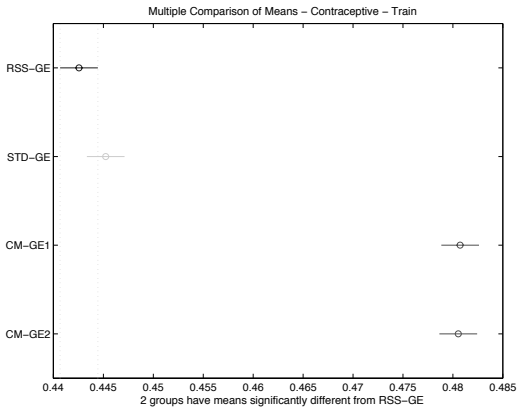
Figure A.11: Direct (GE) comparison of CENS Overall Accuracy performance.



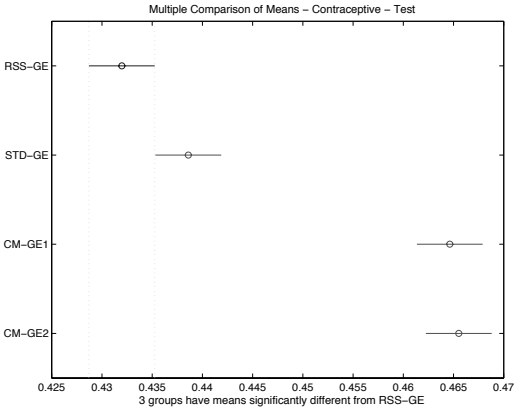
(a) Overall Accuracy (Train)



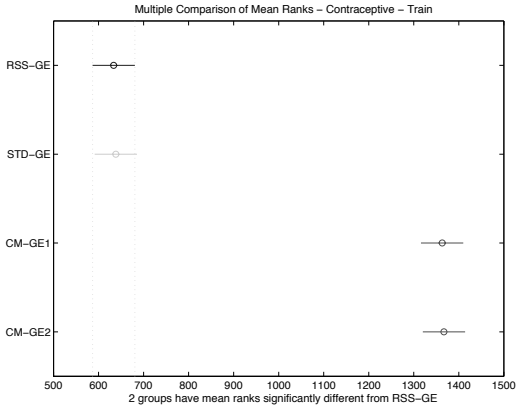
(b) Overall Accuracy (Test)



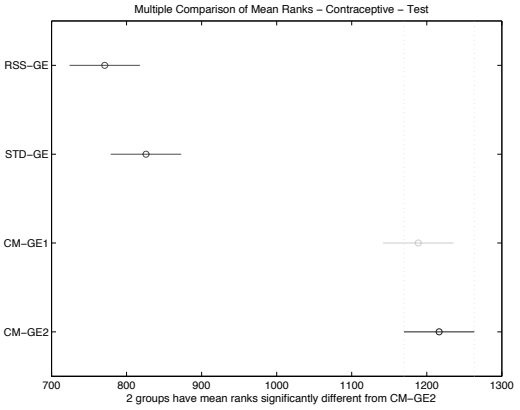
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

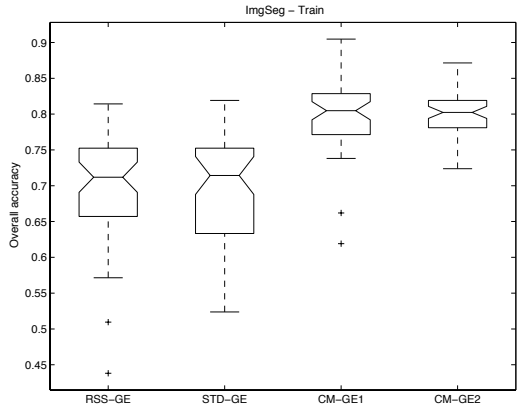


(e) Comparison of Mean Ranks (Train)

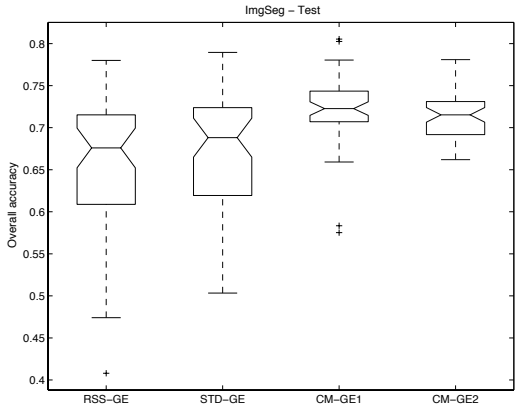


(f) Comparison of Mean Ranks (Test)

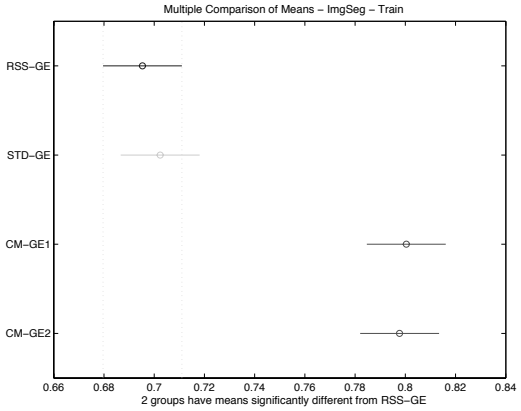
Figure A.12: Direct (GE) comparison of CONT Overall Accuracy performance.



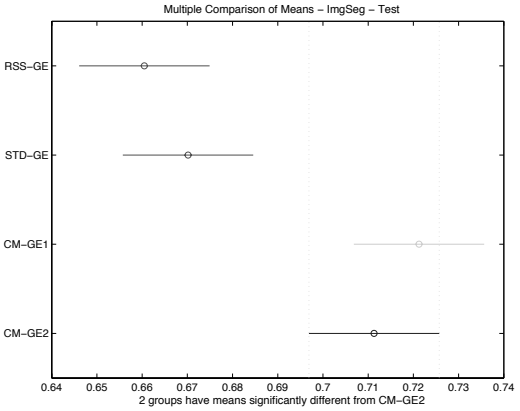
(a) Overall Accuracy (Train)



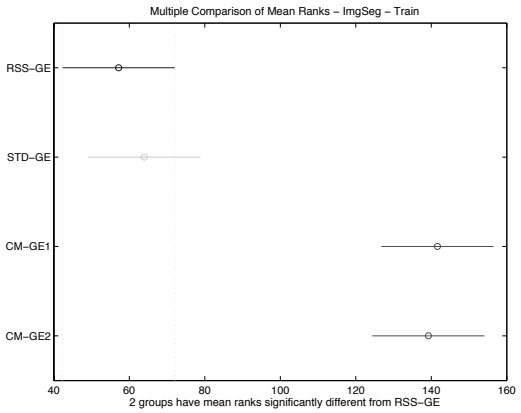
(b) Overall Accuracy (Test)



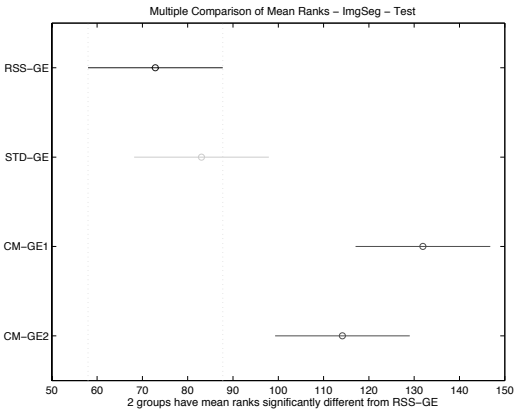
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

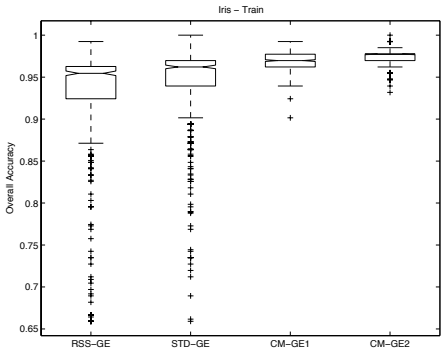


(e) Comparison of Mean Ranks (Train)

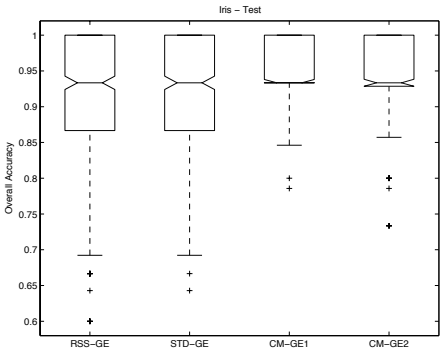


(f) Comparison of Mean Ranks (Test)

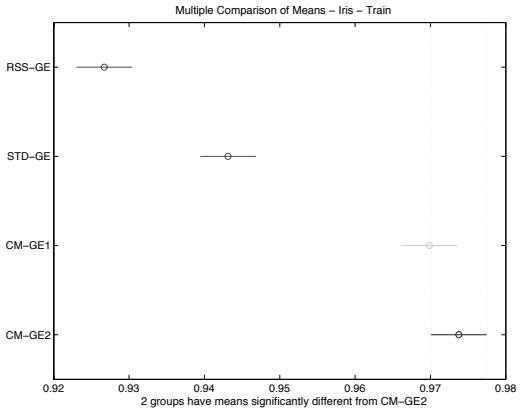
Figure A.13: Direct (GE) comparison of IMAG Overall Accuracy performance.



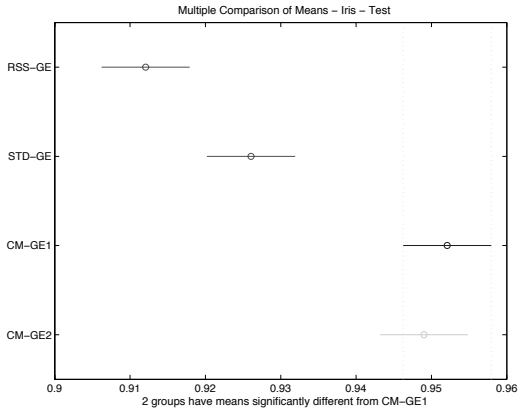
(a) Overall Accuracy (Train)



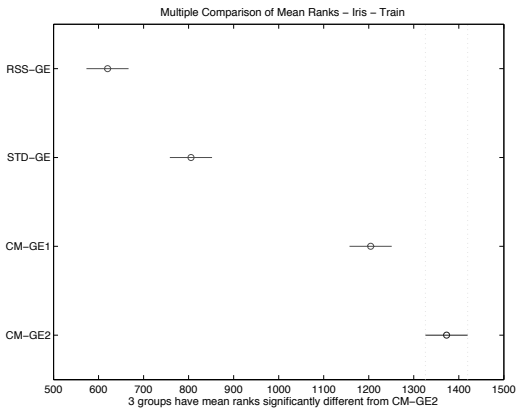
(b) Overall Accuracy (Test)



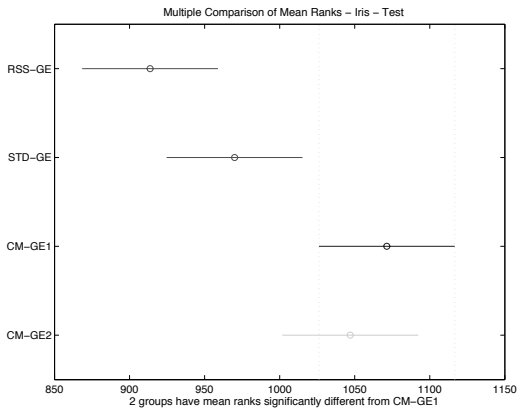
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

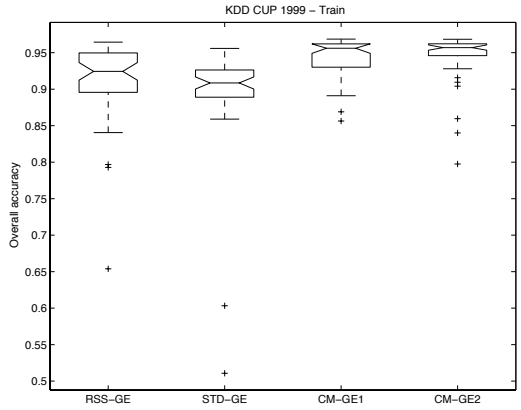


(e) Comparison of Mean Ranks (Train)

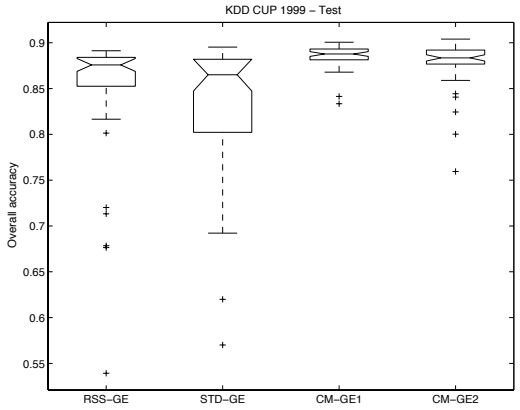


(f) Comparison of Mean Ranks (Test)

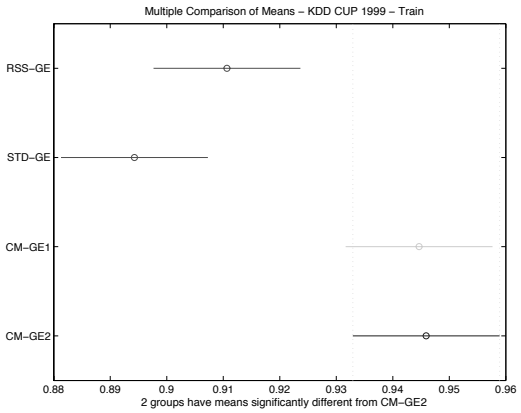
Figure A.14: Direct (GE) comparison of IRIS Overall Accuracy performance.



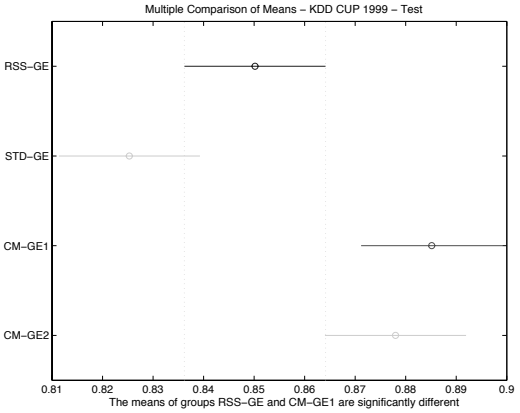
(a) Overall Accuracy (Train)



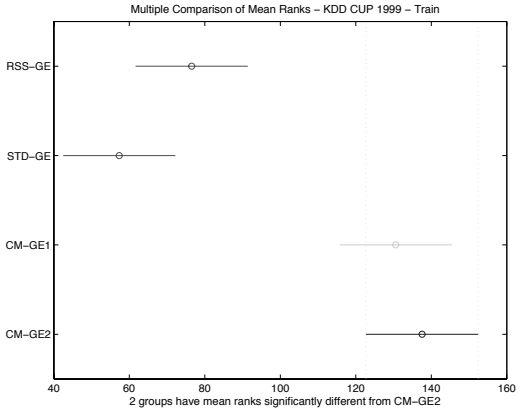
(b) Overall Accuracy (Test)



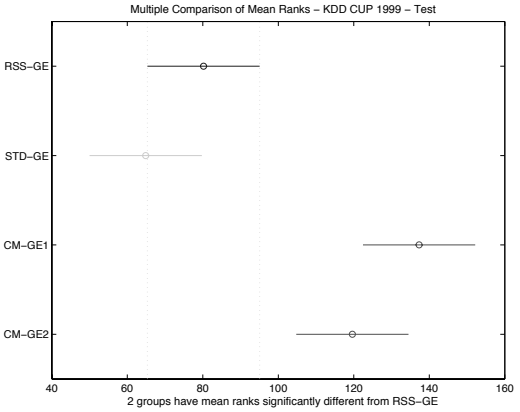
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

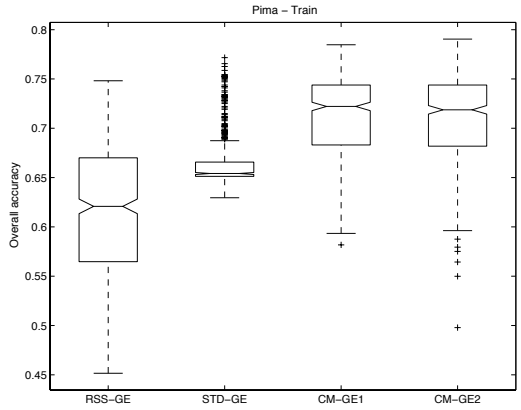


(e) Comparison of Mean Ranks (Train)

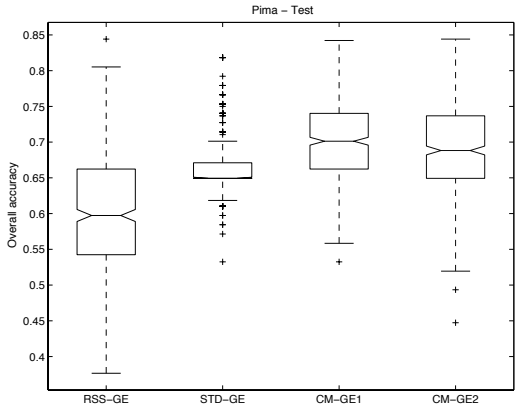


(f) Comparison of Mean Ranks (Test)

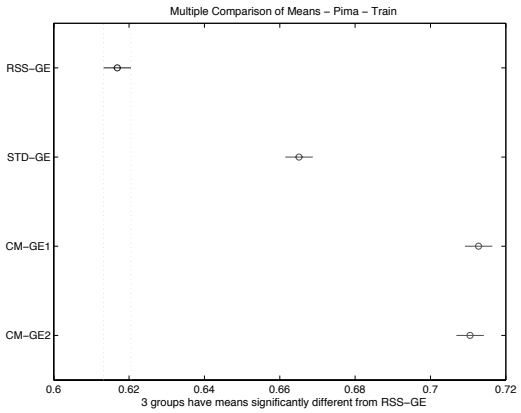
Figure A.15: Direct (GE) comparison of KD99 Overall Accuracy performance.



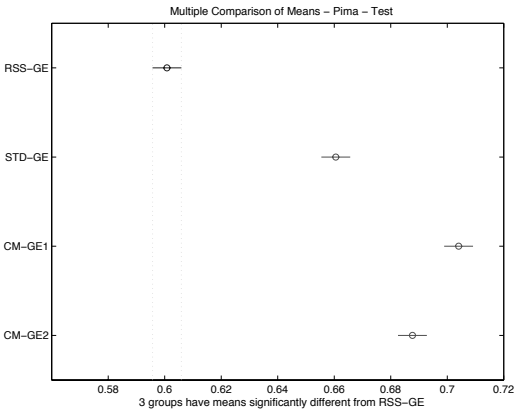
(a) Overall Accuracy (Train)



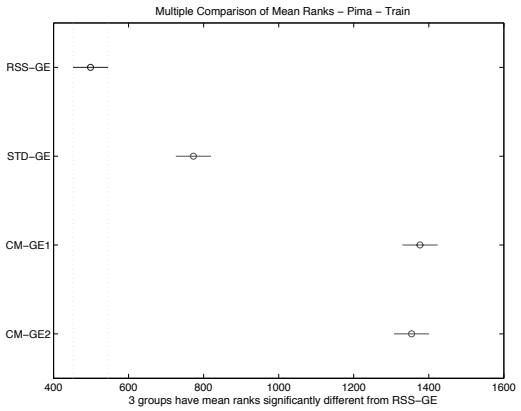
(b) Overall Accuracy (Test)



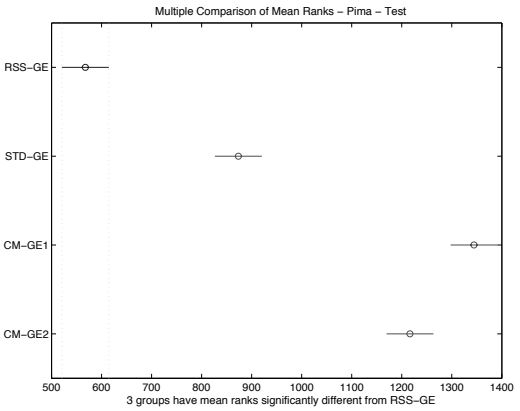
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

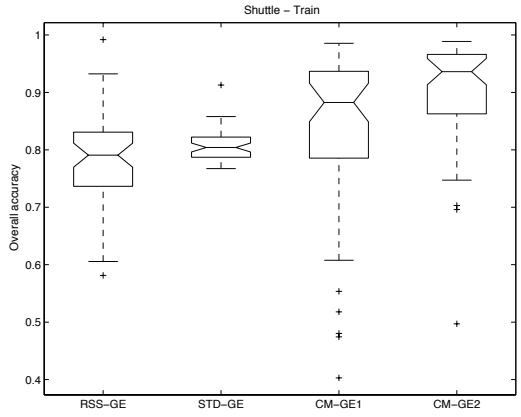


(e) Comparison of Mean Ranks (Train)

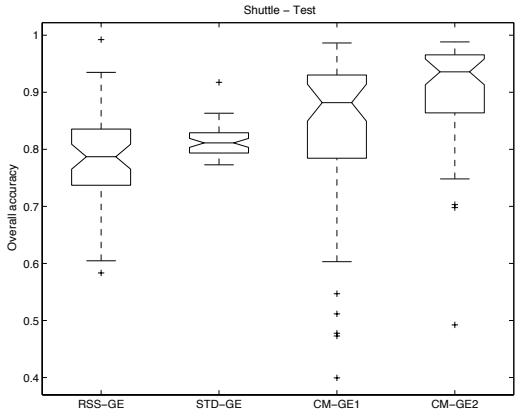


(f) Comparison of Mean Ranks (Test)

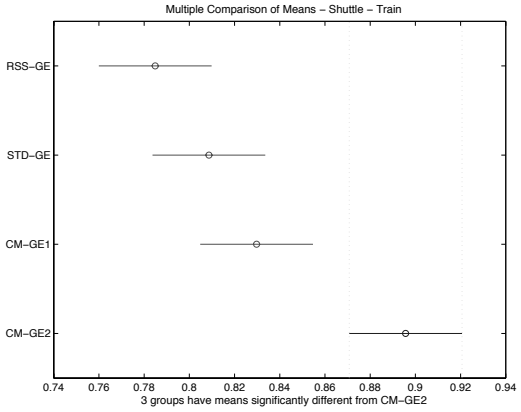
Figure A.16: Direct (GE) comparison of PIMA Overall Accuracy performance.



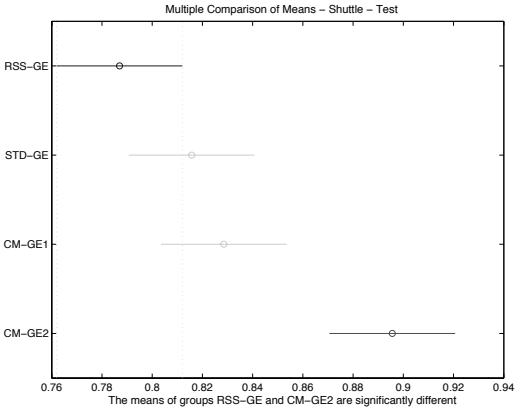
(a) Overall Accuracy (Train)



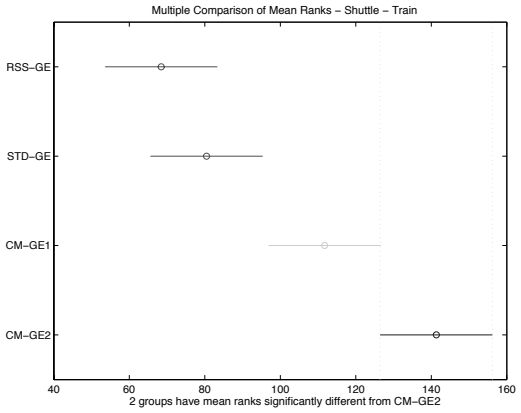
(b) Overall Accuracy (Test)



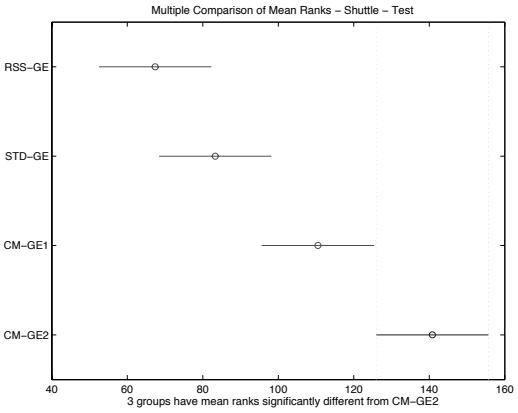
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

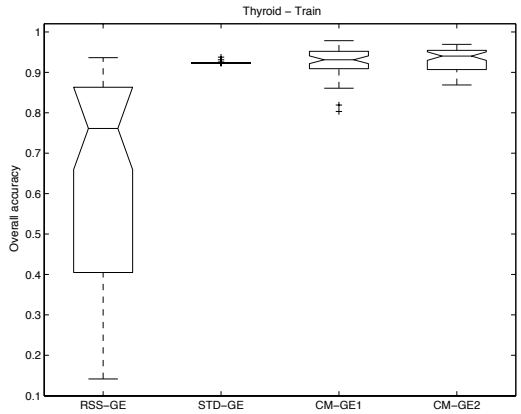


(e) Comparison of Mean Ranks (Train)

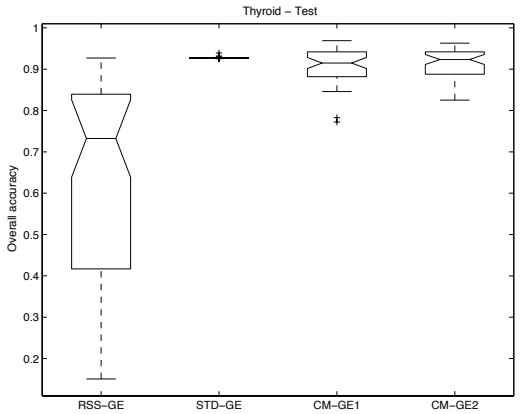


(f) Comparison of Mean Ranks (Test)

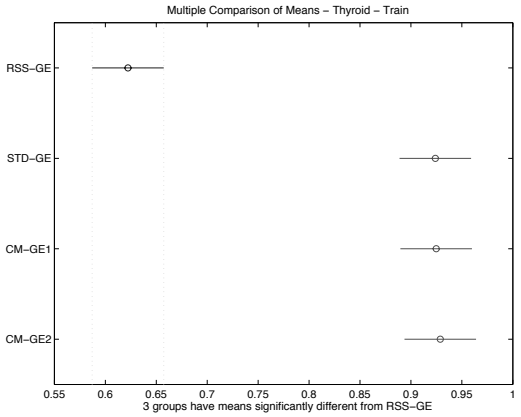
Figure A.17: Direct (GE) comparison of SHUT Overall Accuracy performance.



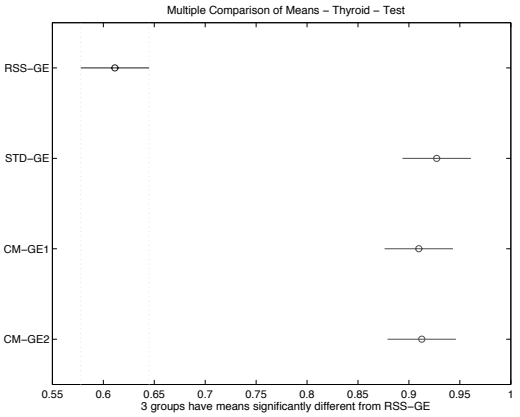
(a) Overall Accuracy (Train)



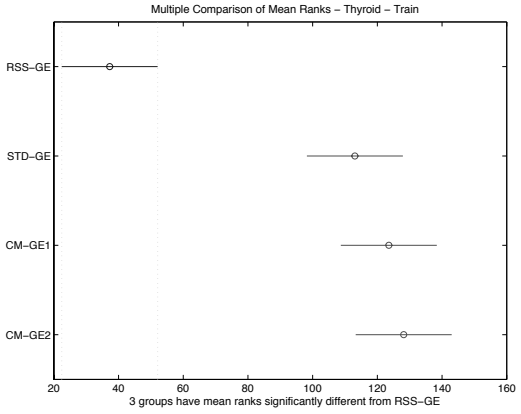
(b) Overall Accuracy (Test)



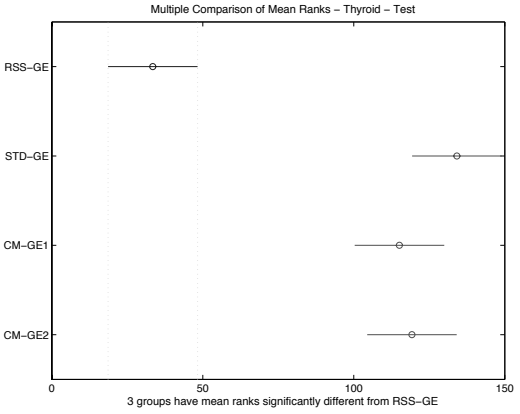
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

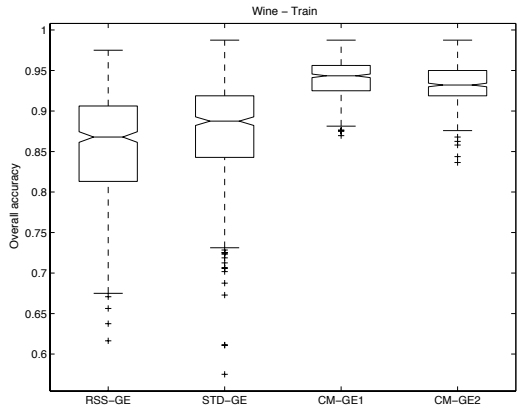


(e) Comparison of Mean Ranks (Train)

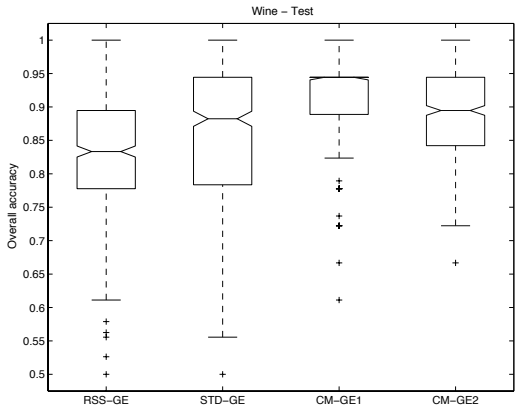


(f) Comparison of Mean Ranks (Test)

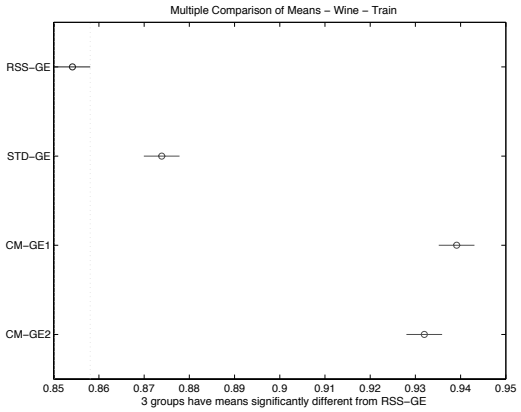
Figure A.18: Direct (GE) comparison of THYD Overall Accuracy performance.



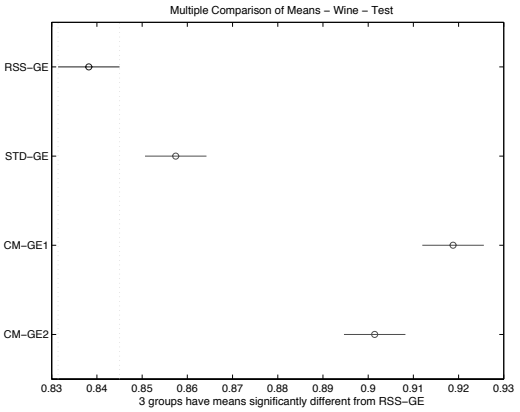
(a) Overall Accuracy (Train)



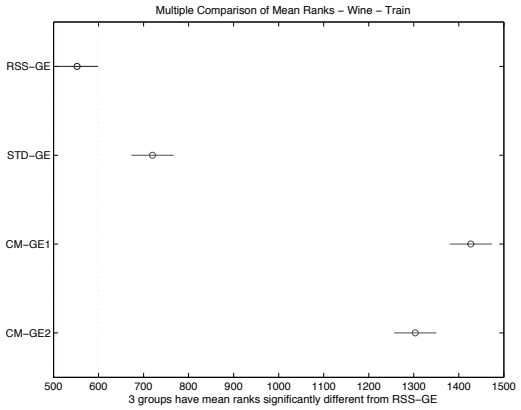
(b) Overall Accuracy (Test)



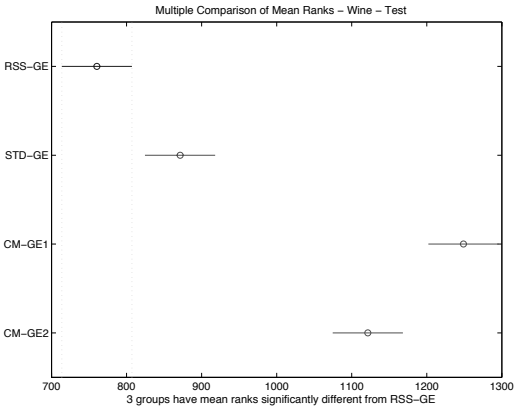
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

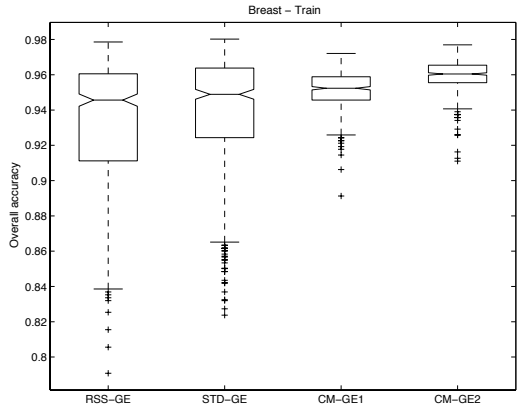


(e) Comparison of Mean Ranks (Train)

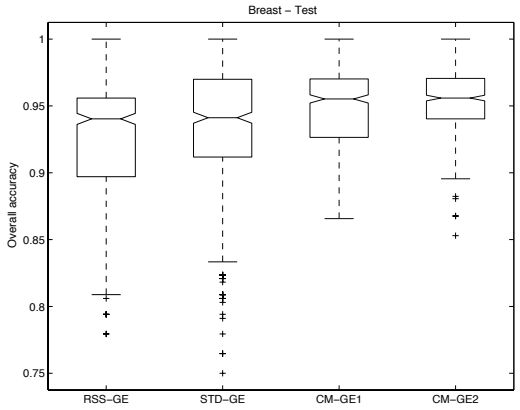


(f) Comparison of Mean Ranks (Test)

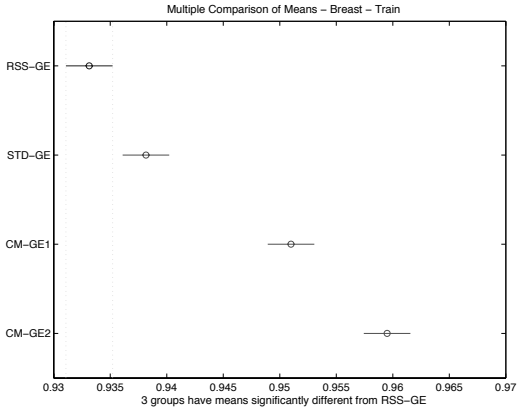
Figure A.19: Direct (GE) comparison of WINE Overall Accuracy performance.



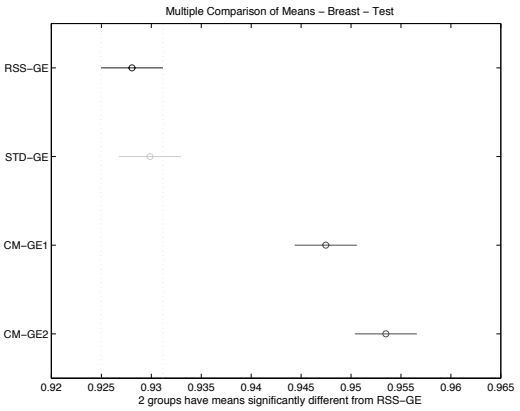
(a) Overall Accuracy (Train)



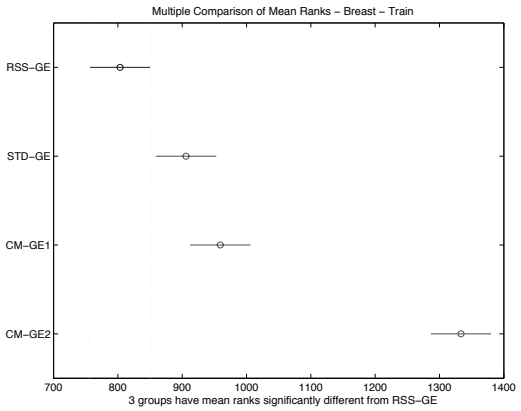
(b) Overall Accuracy (Test)



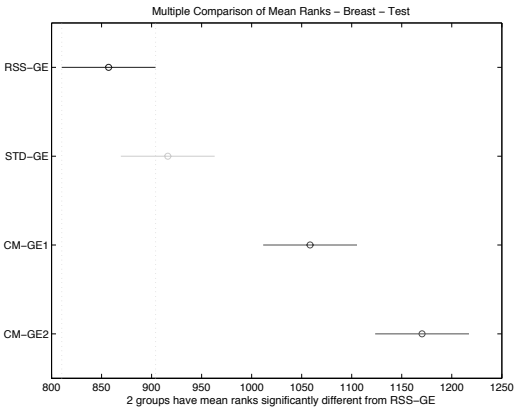
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



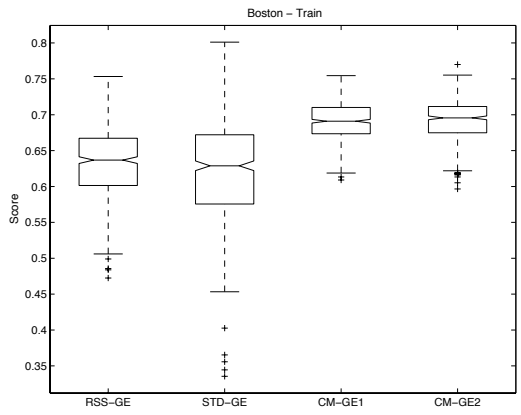
(e) Comparison of Mean Ranks (Train)



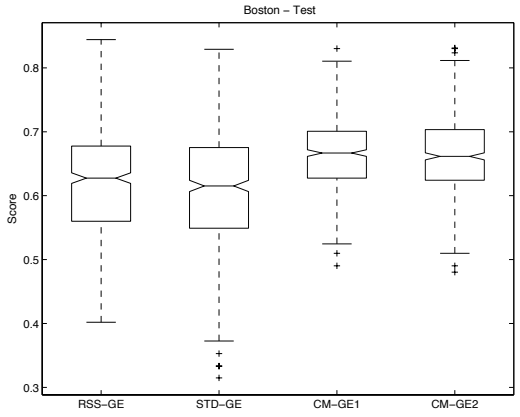
(f) Comparison of Mean Ranks (Test)

Figure A.20: Direct (GE) comparison of WISC Overall Accuracy performance.

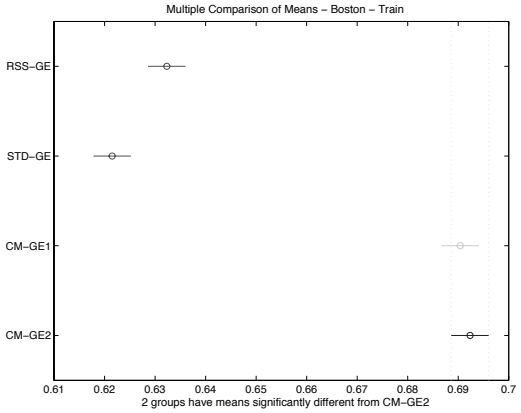
A.2.2 Score



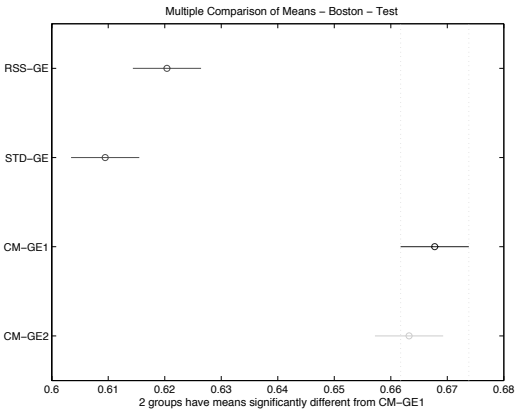
(a) Score (Train)



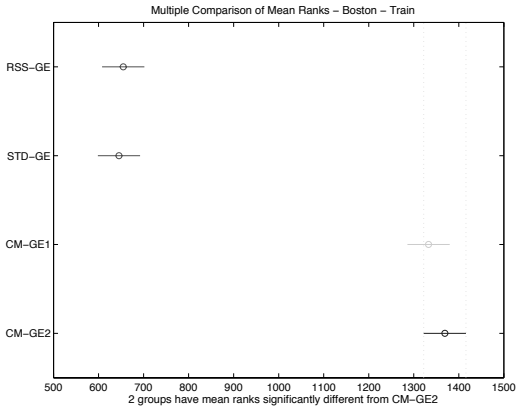
(b) Score (Test)



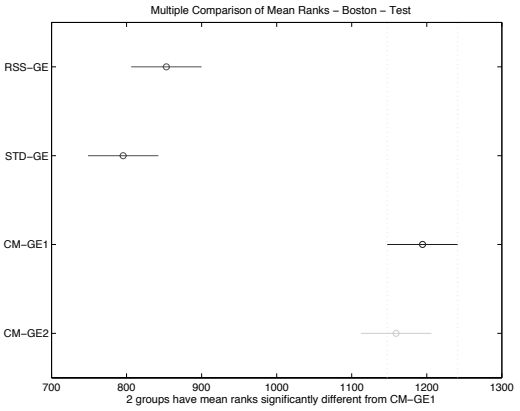
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

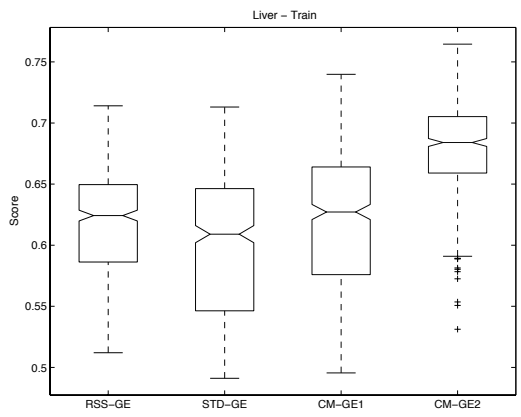


(e) Comparison of Mean Ranks (Train)

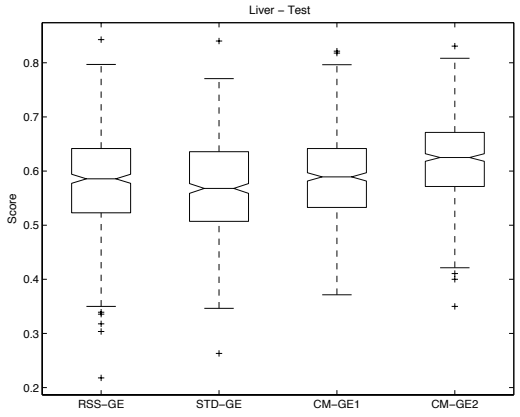


(f) Comparison of Mean Ranks (Test)

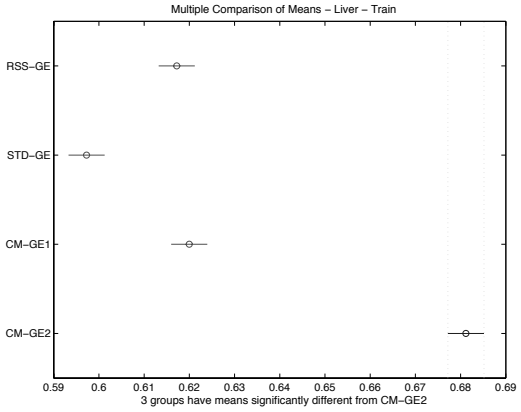
Figure A.21: Direct (GE) comparison of BOST Score performance.



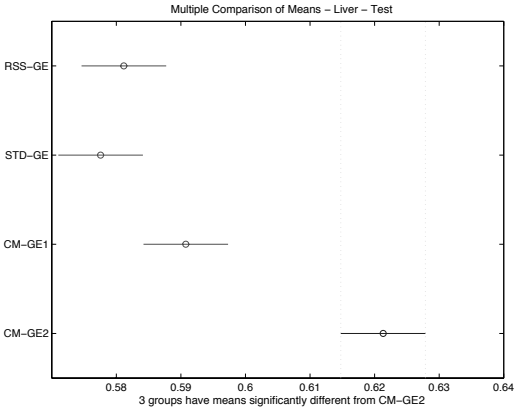
(a) Score (Train)



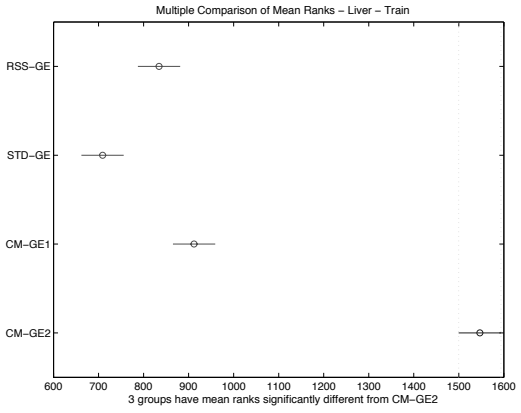
(b) Score (Test)



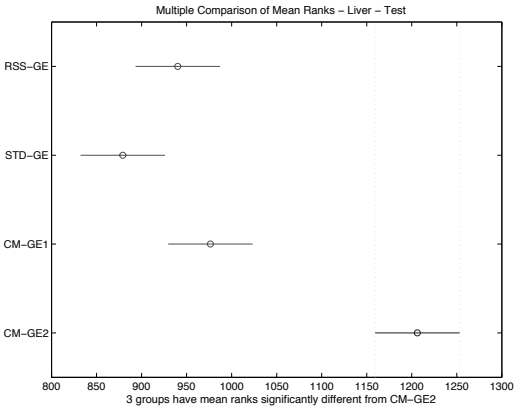
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

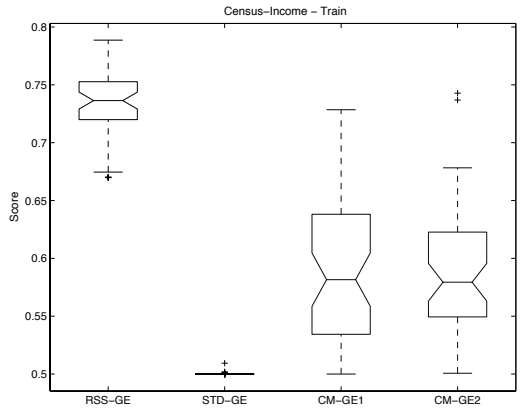


(e) Comparison of Mean Ranks (Train)

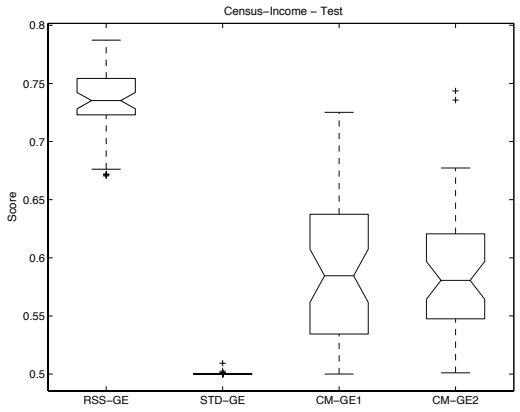


(f) Comparison of Mean Ranks (Test)

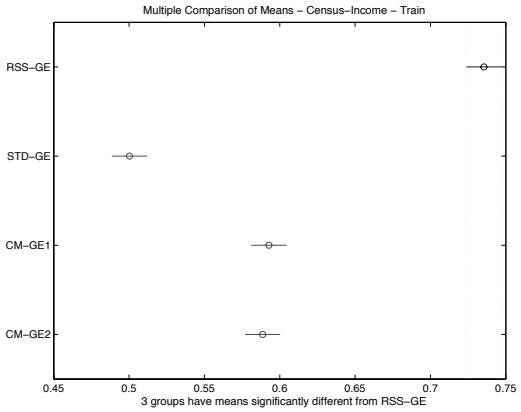
Figure A.22: Direct (GE) comparison of BUPA Score performance.



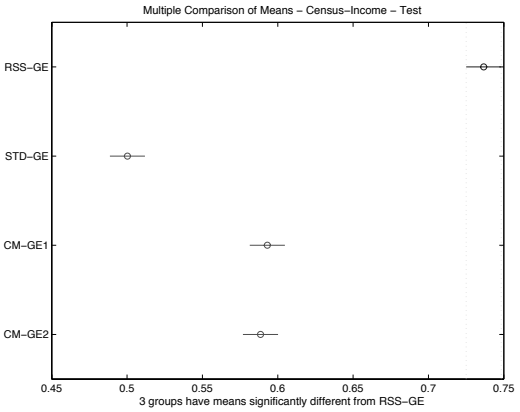
(a) Score (Train)



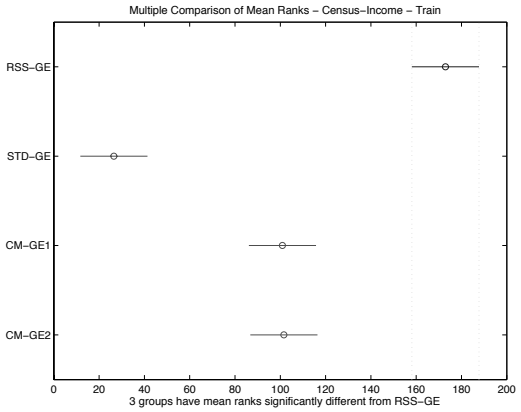
(b) Score (Test)



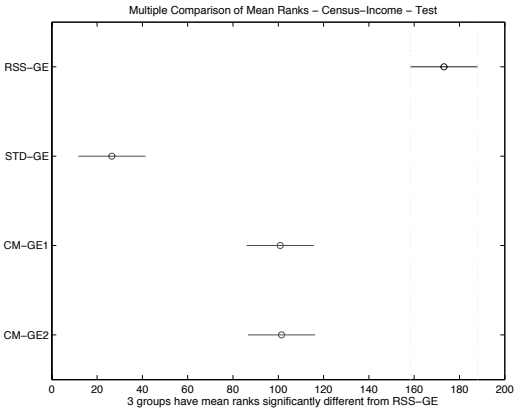
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

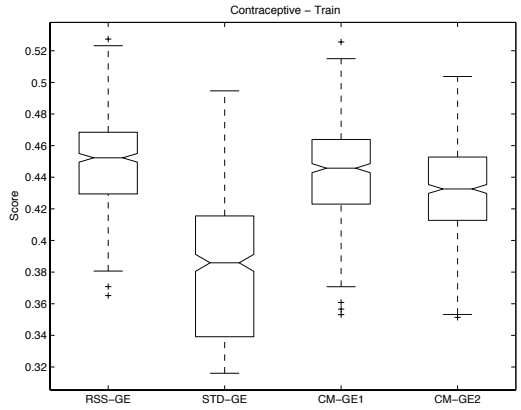


(e) Comparison of Mean Ranks (Train)

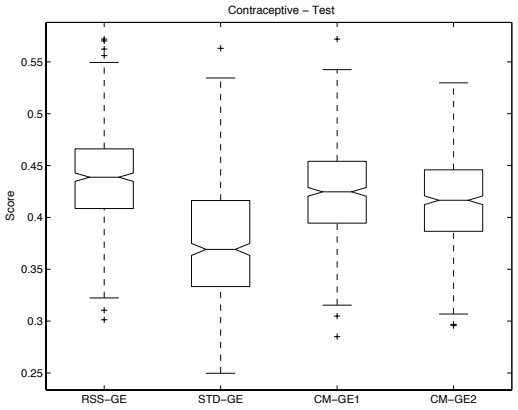


(f) Comparison of Mean Ranks (Test)

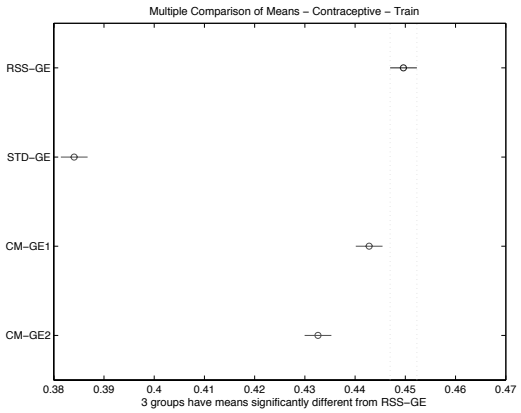
Figure A.23: Direct (GE) comparison of CENS Score performance.



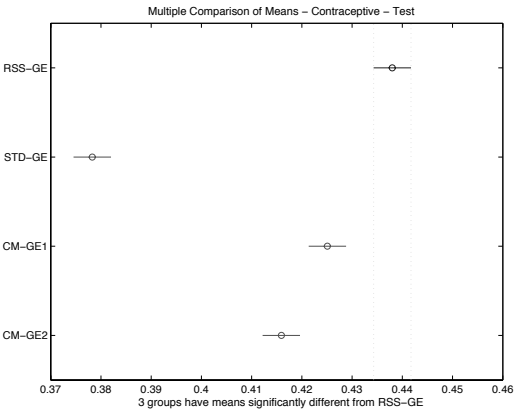
(a) Score (Train)



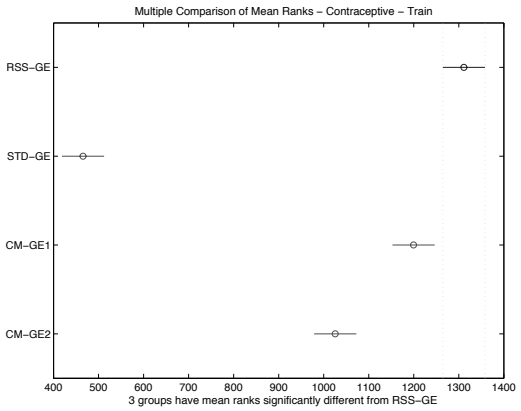
(b) Score (Test)



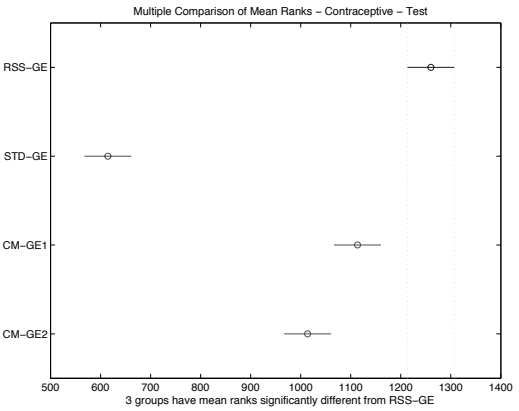
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

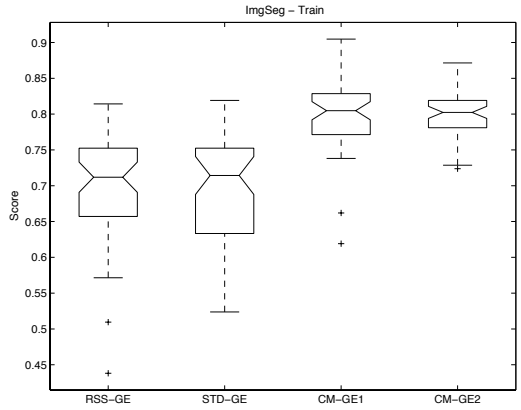


(e) Comparison of Mean Ranks (Train)

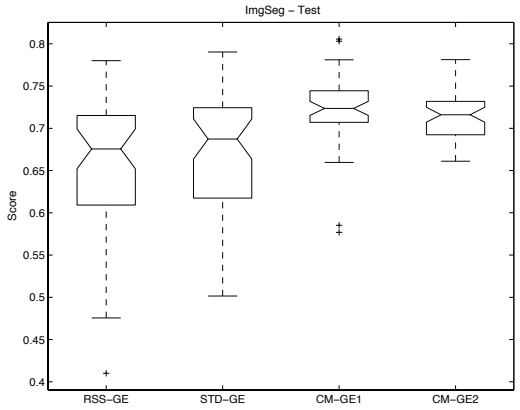


(f) Comparison of Mean Ranks (Test)

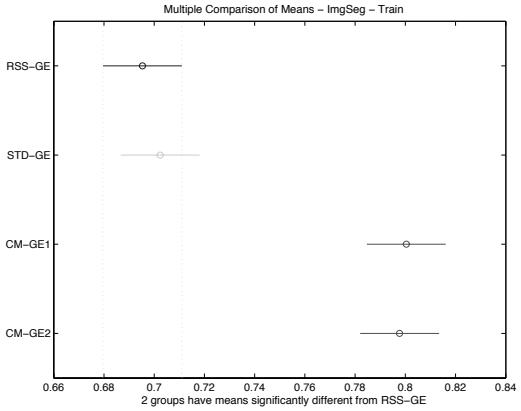
Figure A.24: Direct (GE) comparison of CONT Score performance.



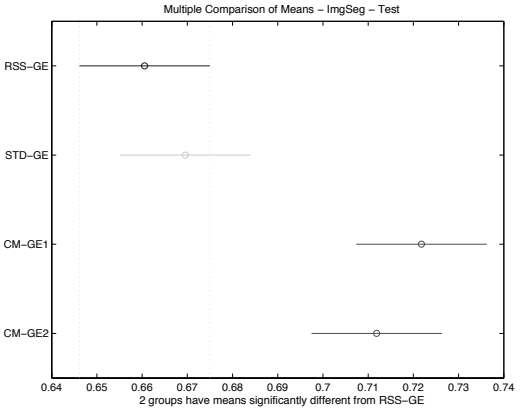
(a) Score (Train)



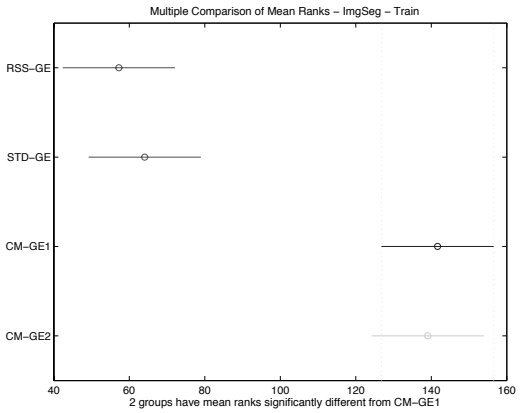
(b) Score (Test)



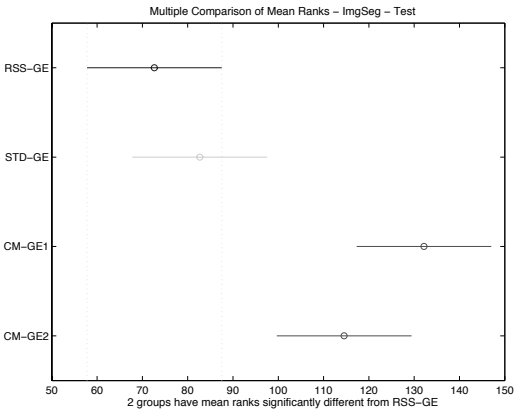
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

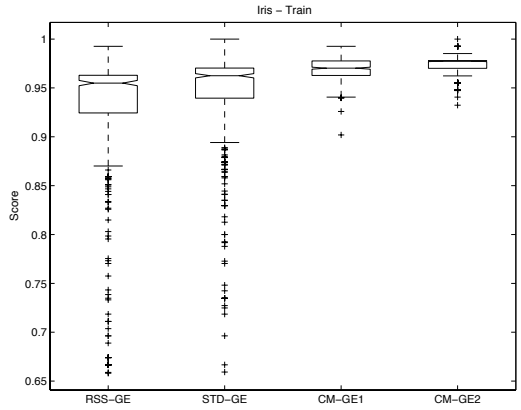


(e) Comparison of Mean Ranks (Train)

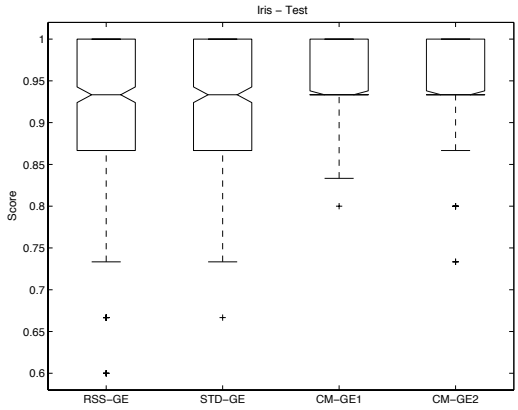


(f) Comparison of Mean Ranks (Test)

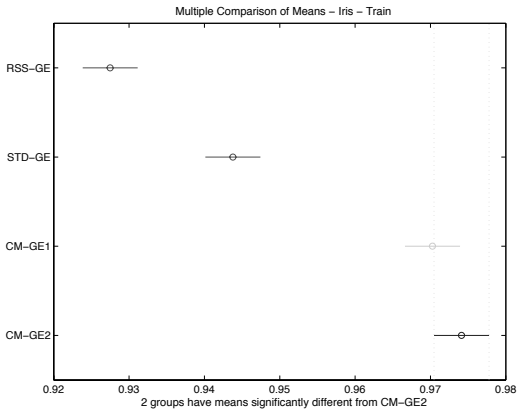
Figure A.25: Direct (GE) comparison of IMAG Score performance.



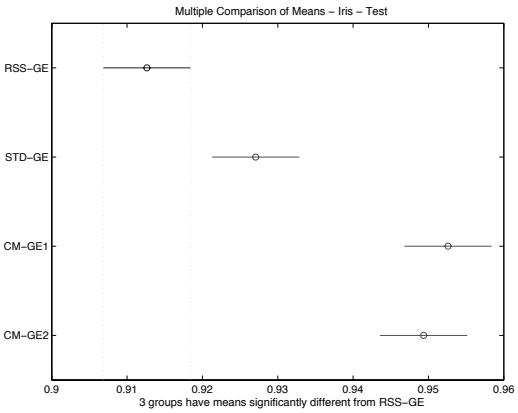
(a) Score (Train)



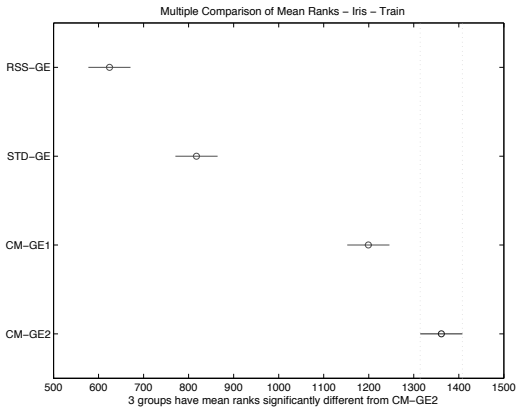
(b) Score (Test)



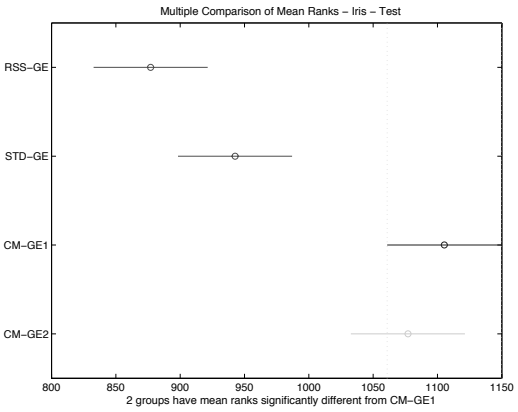
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

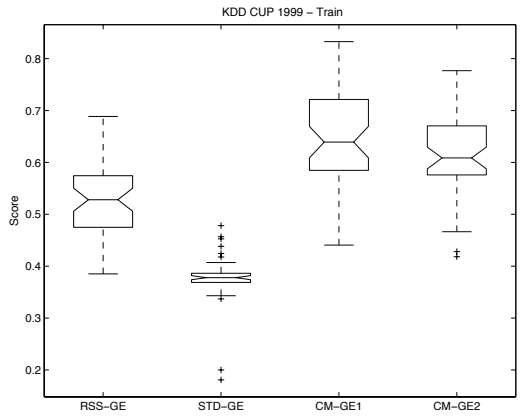


(e) Comparison of Mean Ranks (Train)

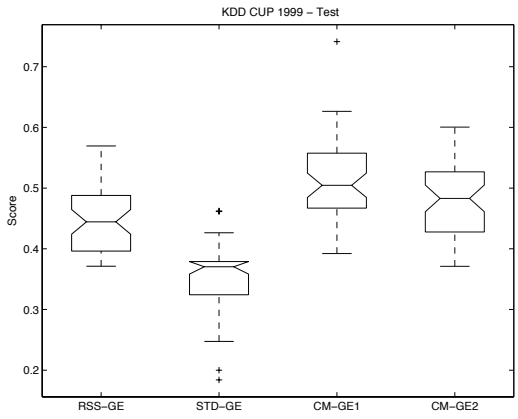


(f) Comparison of Mean Ranks (Test)

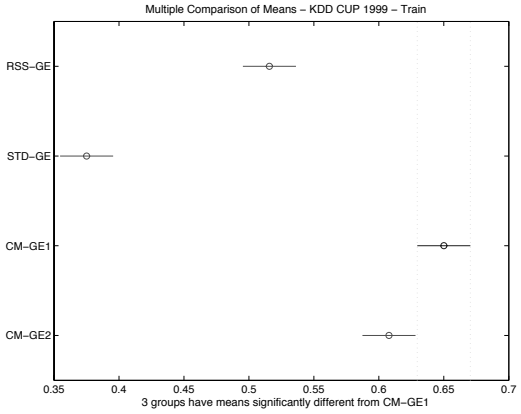
Figure A.26: Direct (GE) comparison of IRIS Score performance.



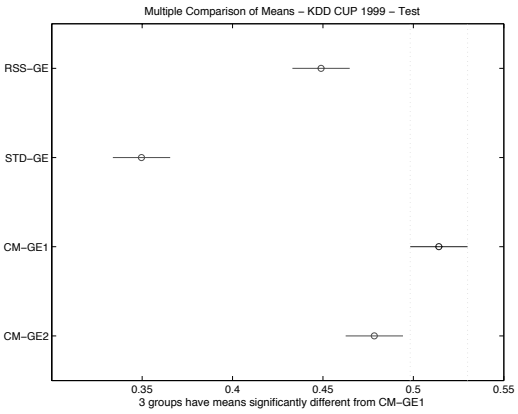
(a) Score (Train)



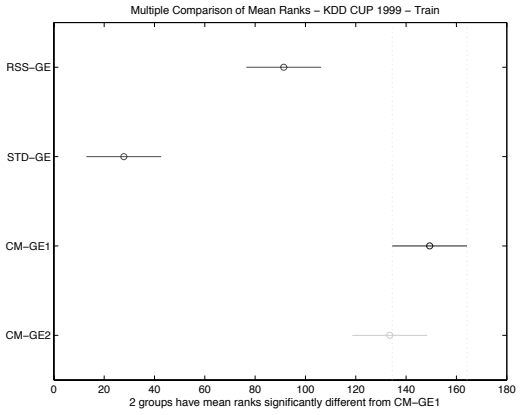
(b) Score (Test)



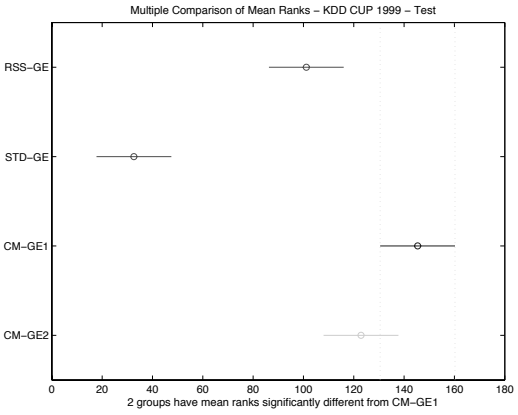
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

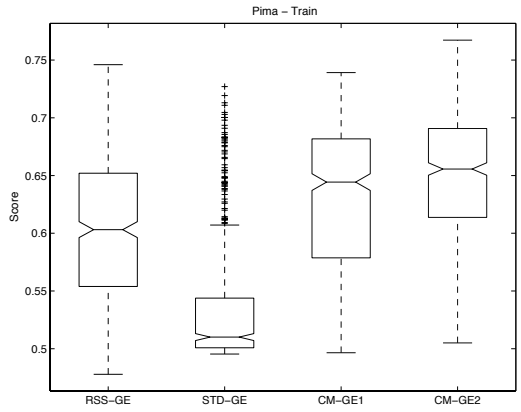


(e) Comparison of Mean Ranks (Train)

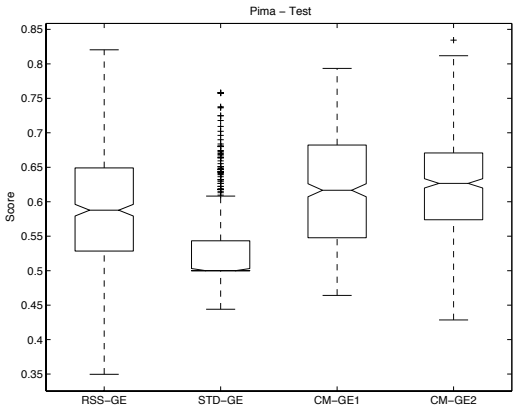


(f) Comparison of Mean Ranks (Test)

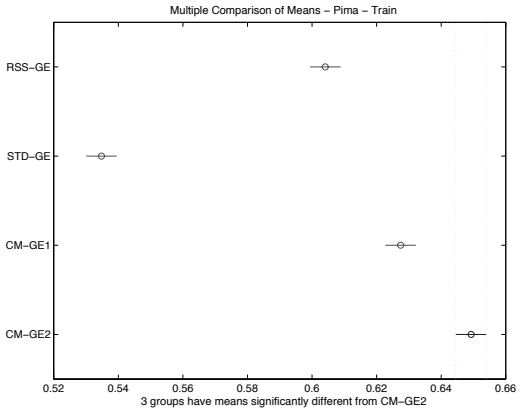
Figure A.27: Direct (GE) comparison of KD99 Score performance.



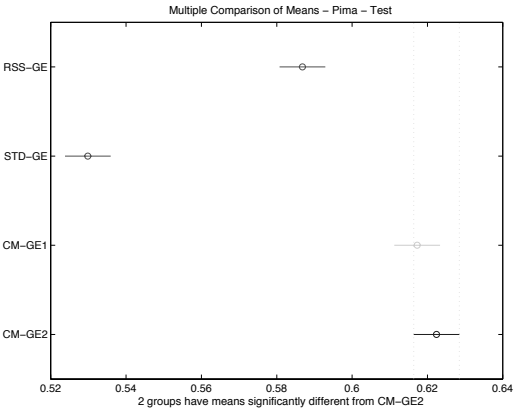
(a) Score (Train)



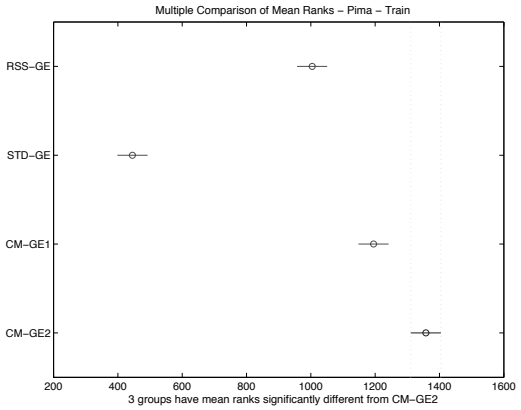
(b) Score (Test)



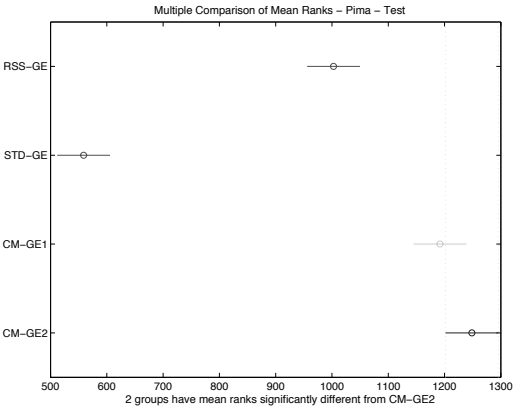
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

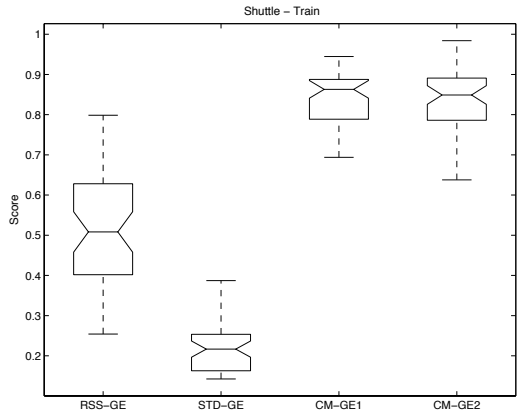


(e) Comparison of Mean Ranks (Train)

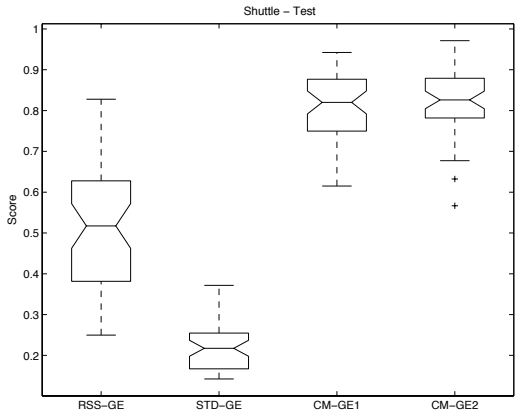


(f) Comparison of Mean Ranks (Test)

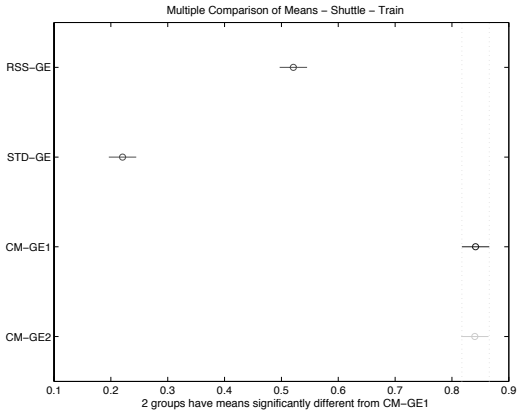
Figure A.28: Direct (GE) comparison of PIMA Score performance.



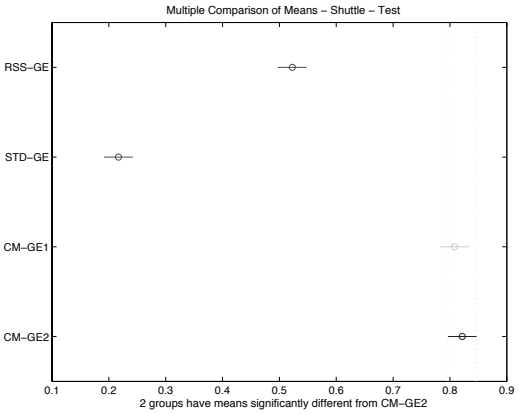
(a) Score (Train)



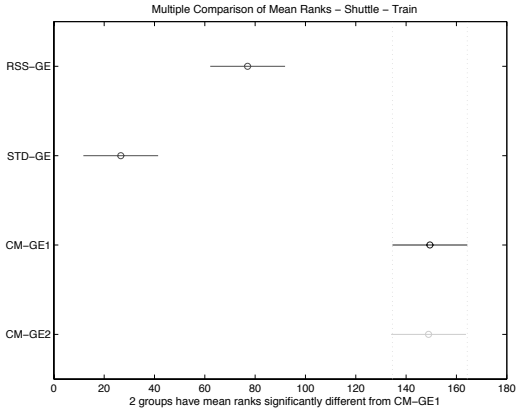
(b) Score (Test)



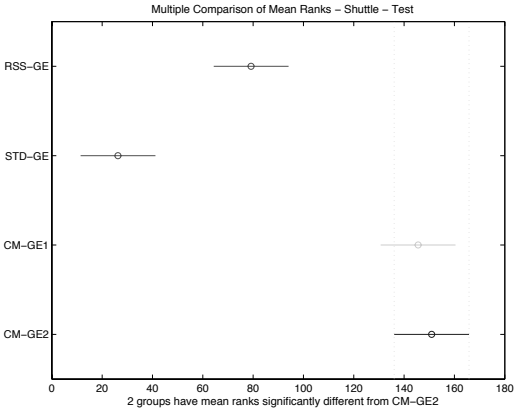
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

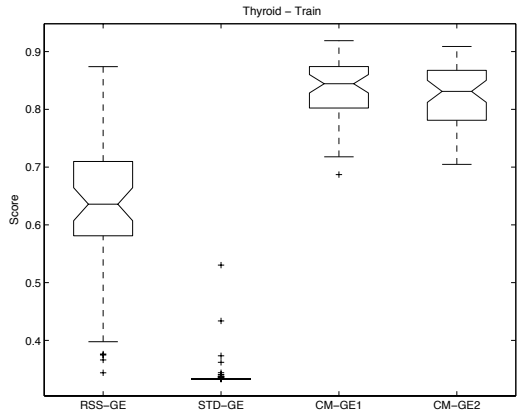


(e) Comparison of Mean Ranks (Train)

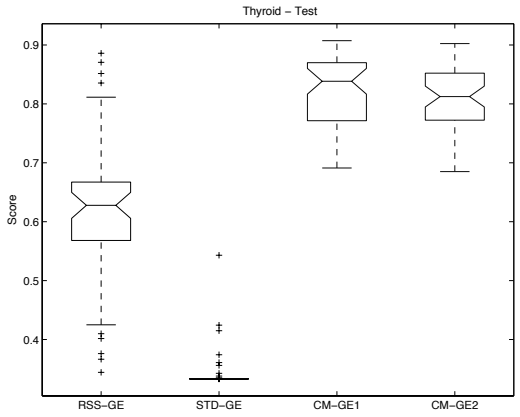


(f) Comparison of Mean Ranks (Test)

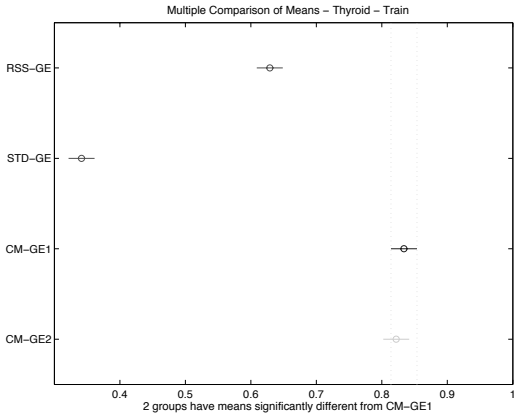
Figure A.29: Direct (GE) comparison of SHUT Score performance.



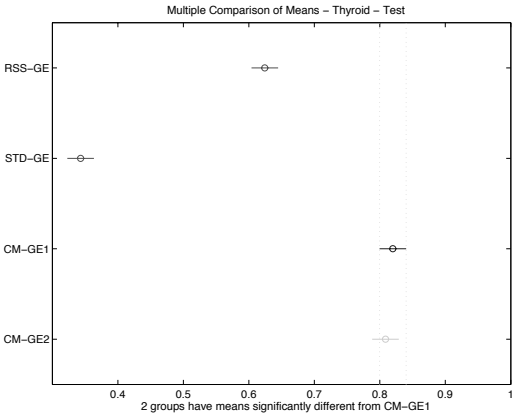
(a) Score (Train)



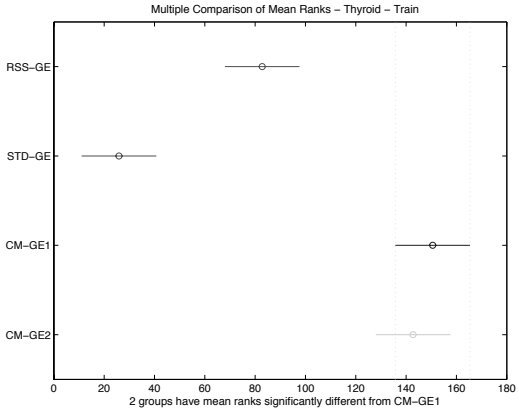
(b) Score (Test)



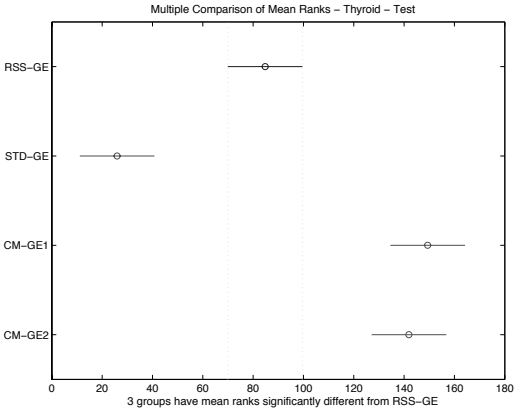
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

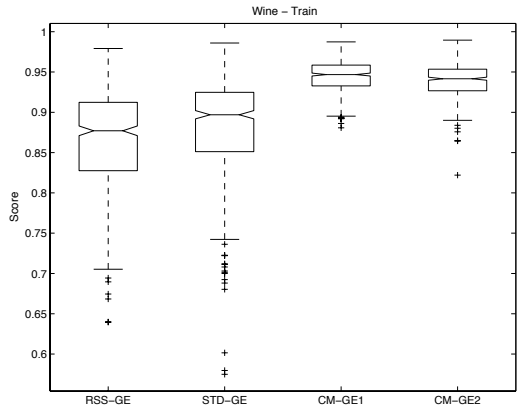


(e) Comparison of Mean Ranks (Train)

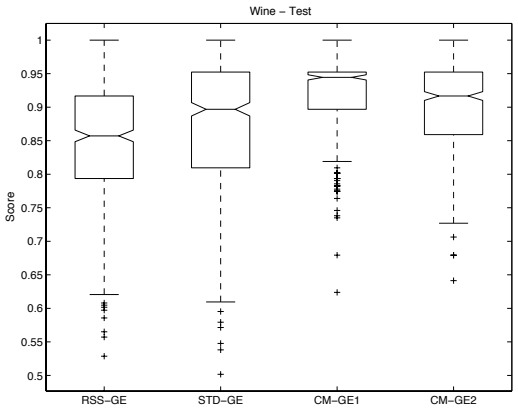


(f) Comparison of Mean Ranks (Test)

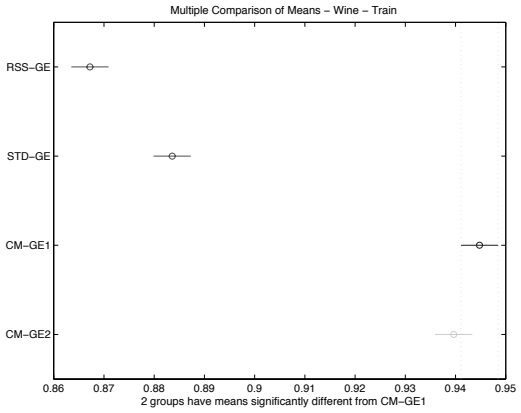
Figure A.30: Direct (GE) comparison of THYD Score performance.



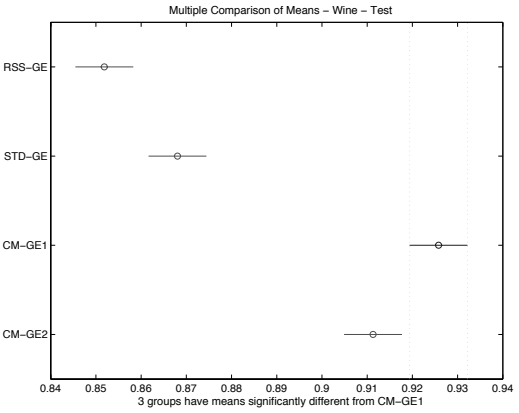
(a) Score (Train)



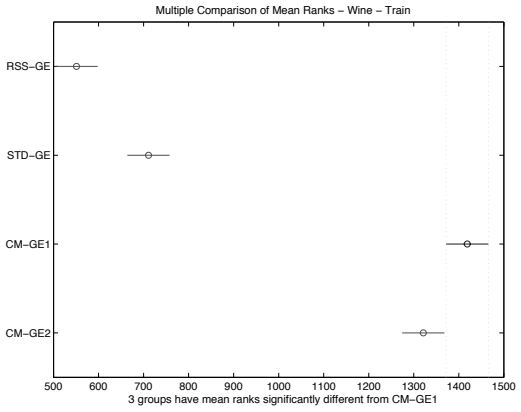
(b) Score (Test)



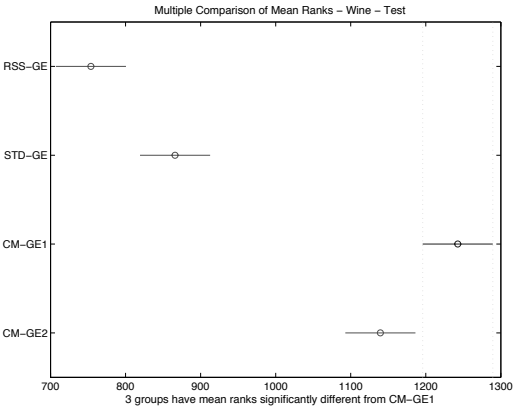
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

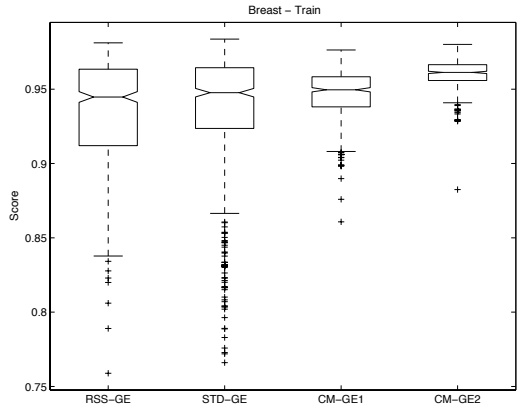


(e) Comparison of Mean Ranks (Train)

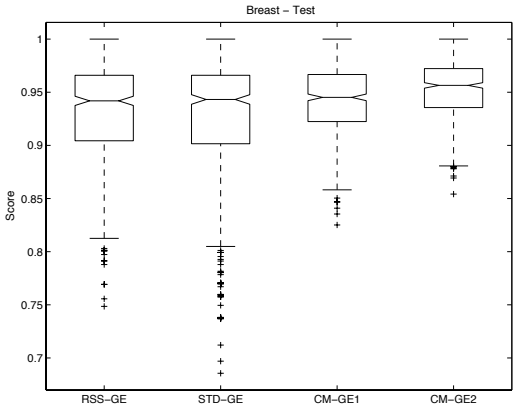


(f) Comparison of Mean Ranks (Test)

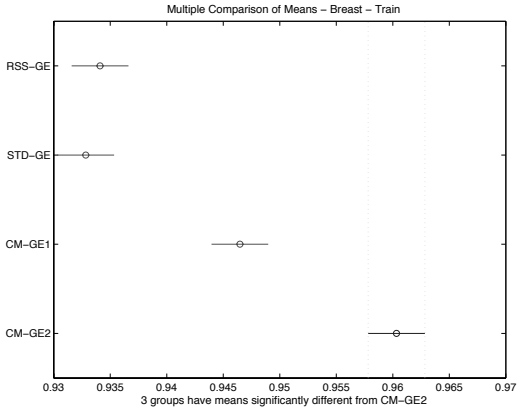
Figure A.31: Direct (GE) comparison of WINE Score performance.



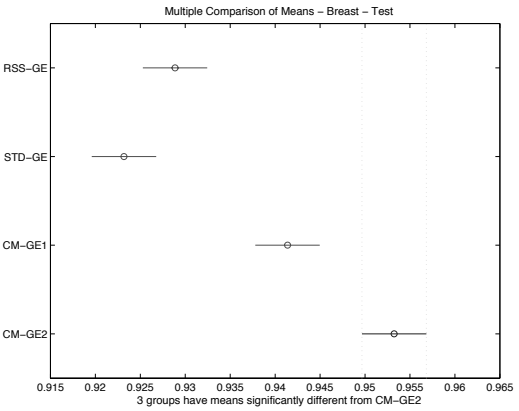
(a) Score (Train)



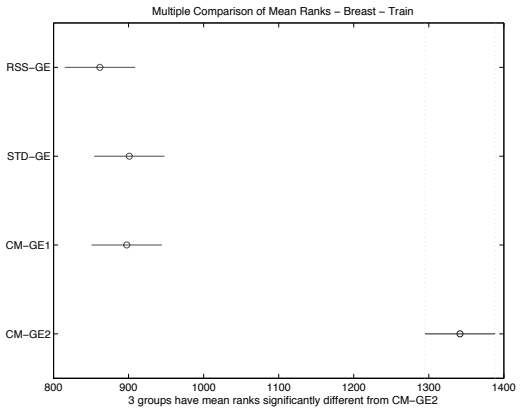
(b) Score (Test)



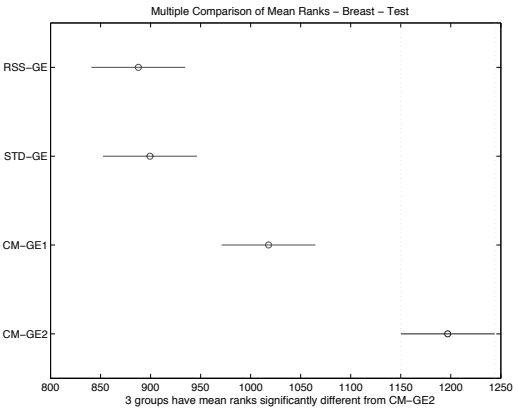
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



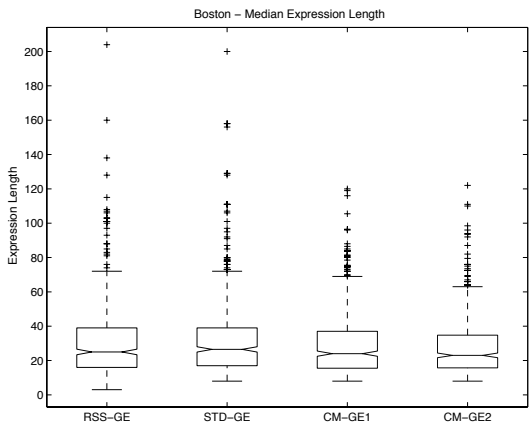
(e) Comparison of Mean Ranks (Train)



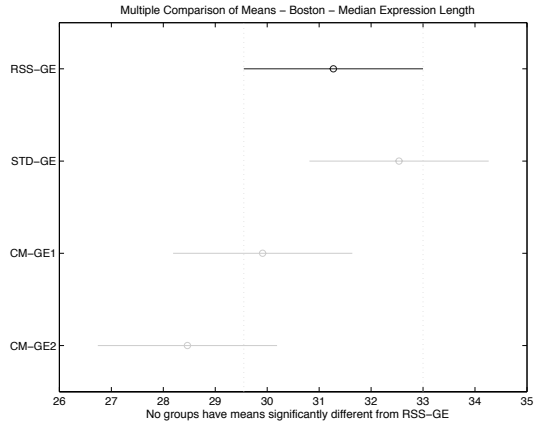
(f) Comparison of Mean Ranks (Test)

Figure A.32: Direct (GE) comparison of WISC Score performance.

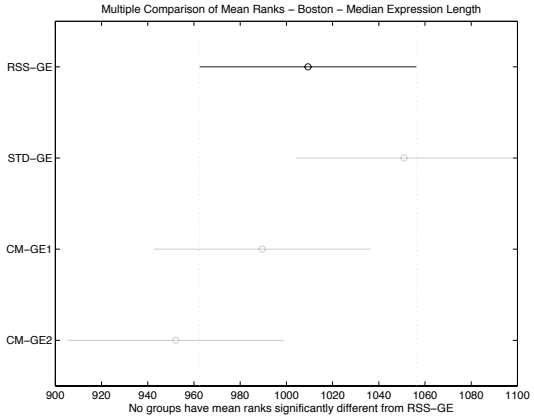
A.2.3 Solution Complexity (String length)



(a) Solution Length (strlen)

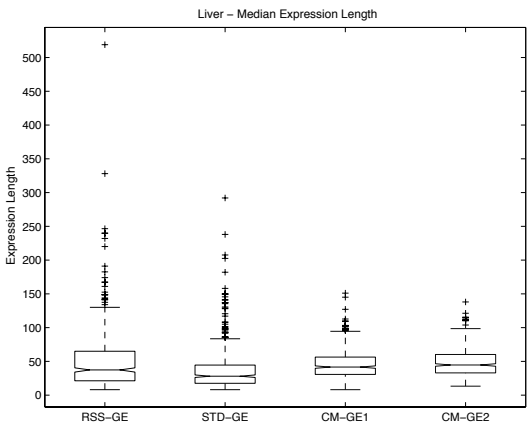


(b) Comparison of Means

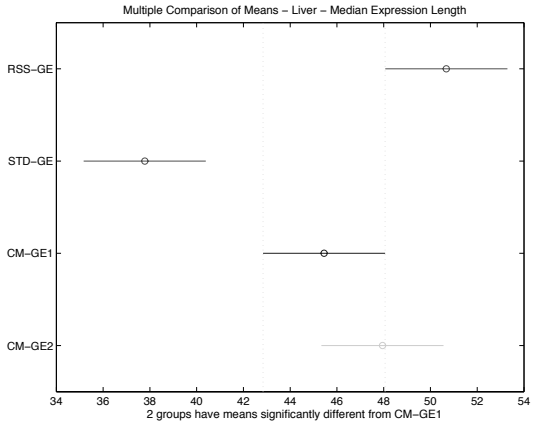


(c) Comparison of Mean Ranks

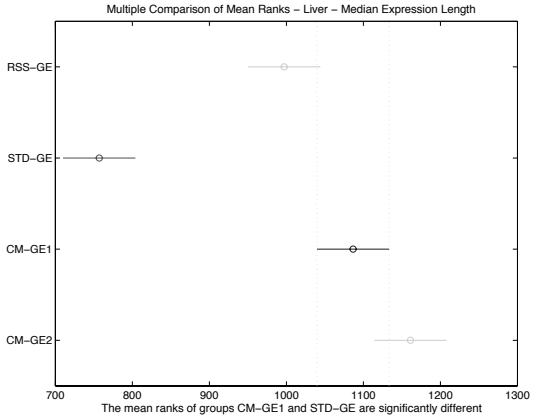
Figure A.33: Direct (GE) comparison of solution length on BOST.



(a) Solution Length (strlen)

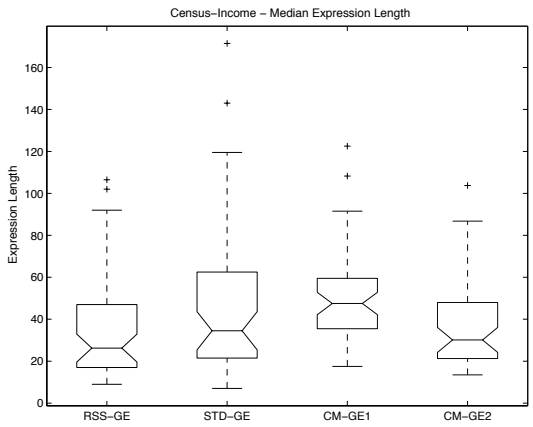


(b) Comparison of Means

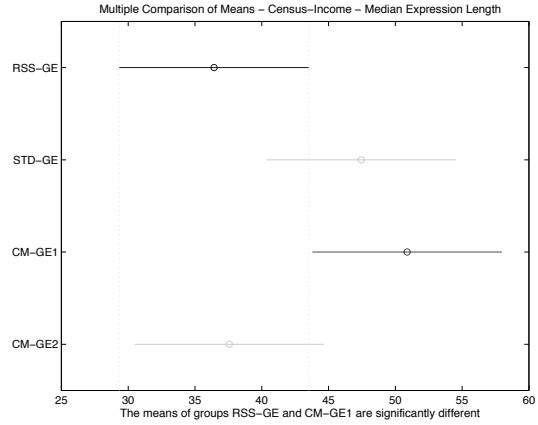


(c) Comparison of Mean Ranks

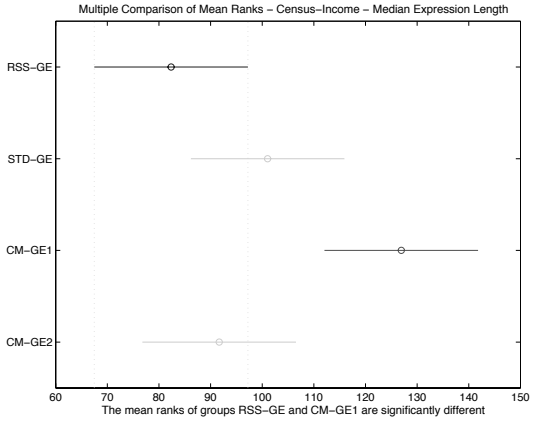
Figure A.34: Direct (GE) comparison of solution length on BUPA.



(a) Solution Length (strlen)

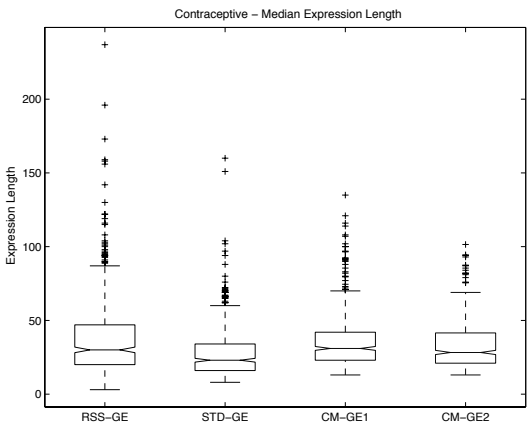


(b) Comparison of Means

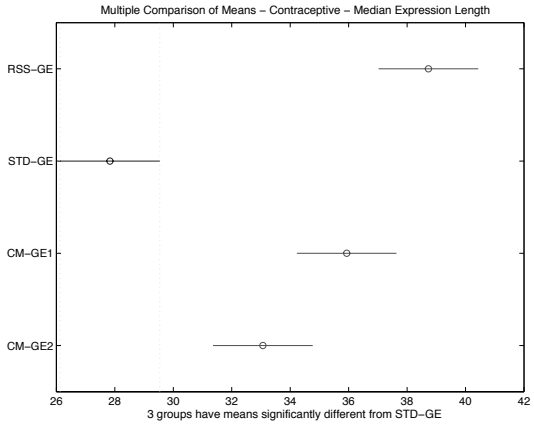


(c) Comparison of Mean Ranks

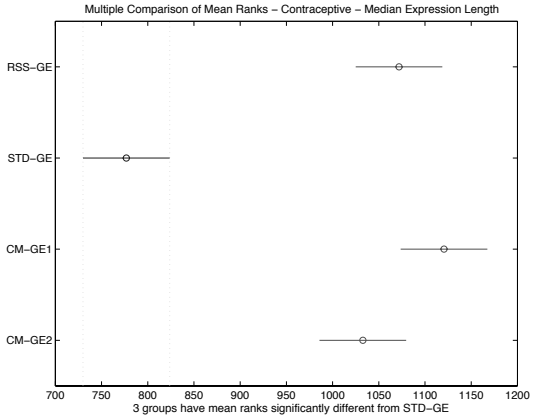
Figure A.35: Direct (GE) comparison of solution length on CENS.



(a) Solution Length (strlen)

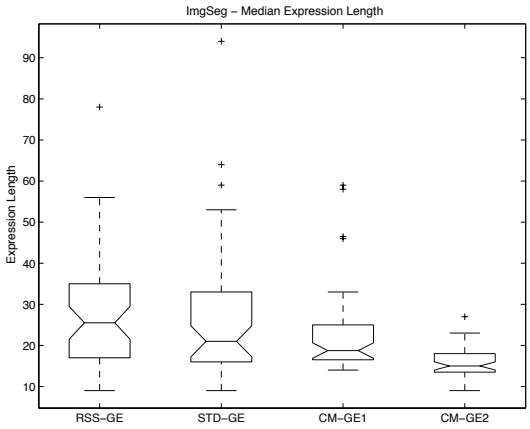


(b) Comparison of Means

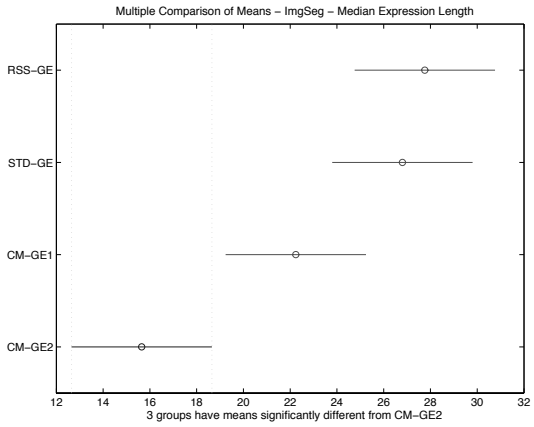


(c) Comparison of Mean Ranks

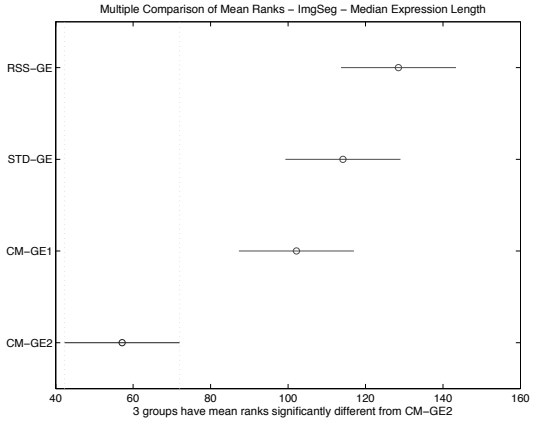
Figure A.36: Direct (GE) comparison of solution length on CONT.



(a) Solution Length (strlen)

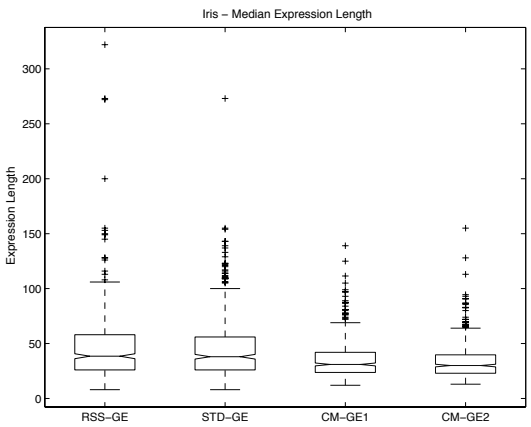


(b) Comparison of Means

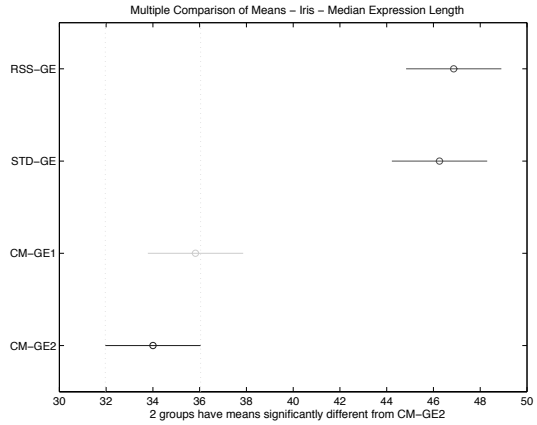


(c) Comparison of Mean Ranks

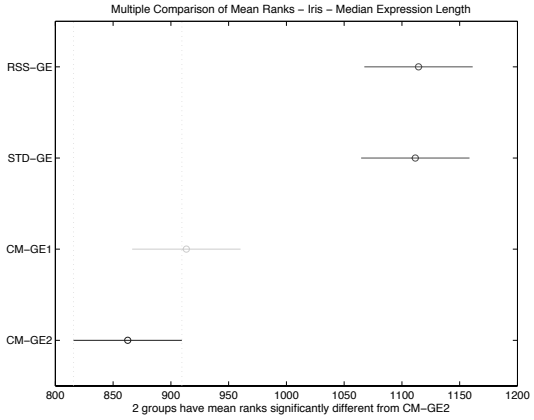
Figure A.37: Direct (GE) comparison of solution length on IMAG.



(a) Solution Length (strlen)

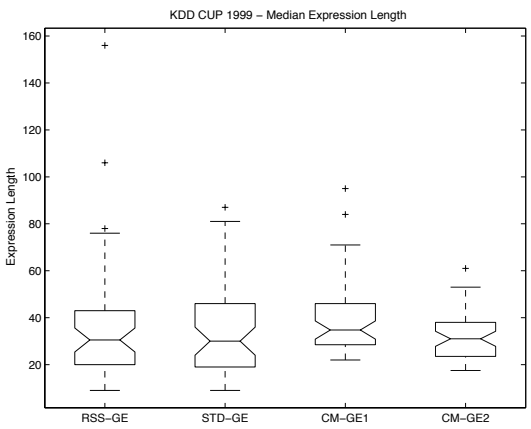


(b) Comparison of Means

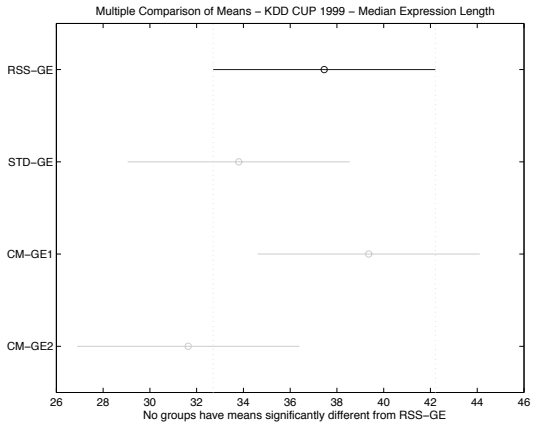


(c) Comparison of Mean Ranks

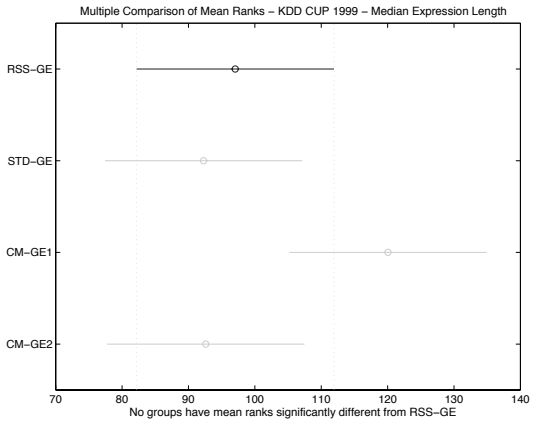
Figure A.38: Direct (GE) comparison of solution length on IRIS.



(a) Solution Length (strlen)

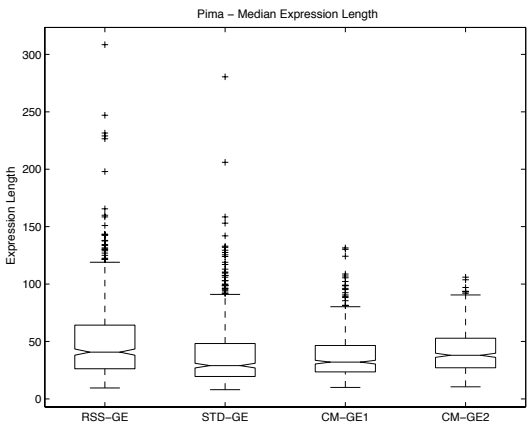


(b) Comparison of Means

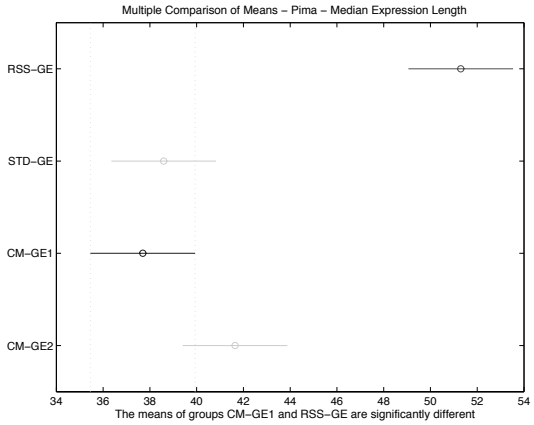


(c) Comparison of Mean Ranks

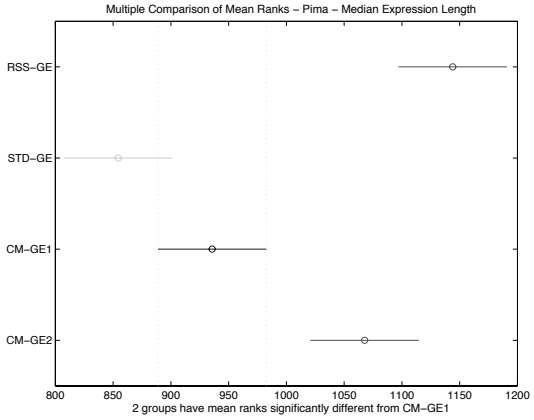
Figure A.39: Direct (GE) comparison of solution length on KD99.



(a) Solution Length (strlen)

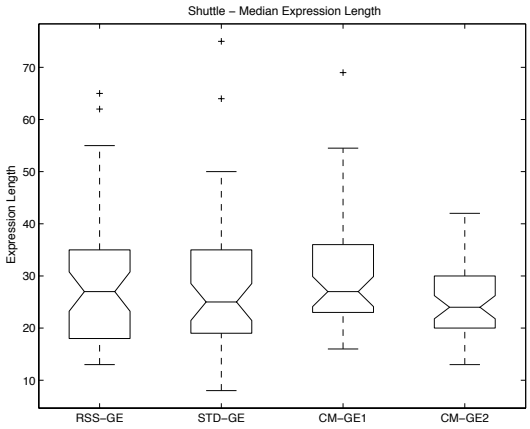


(b) Comparison of Means

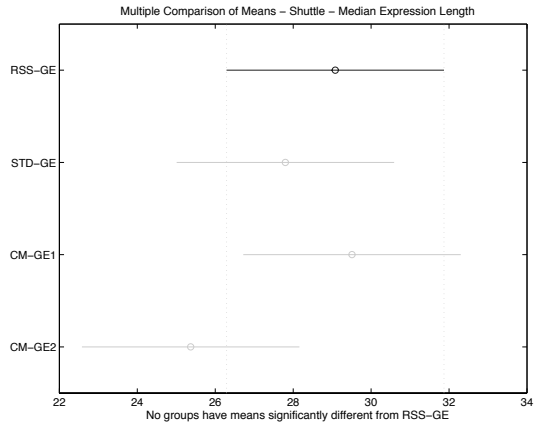


(c) Comparison of Mean Ranks

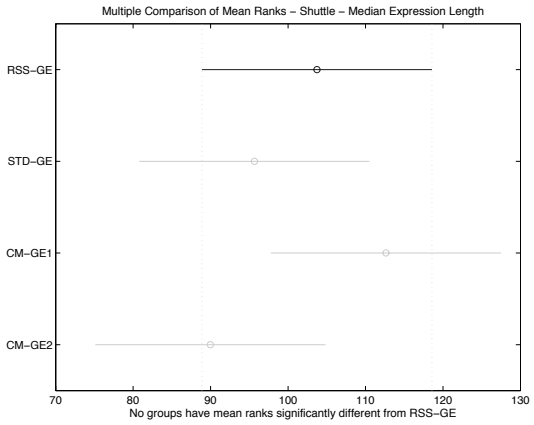
Figure A.40: Direct (GE) comparison of solution length on PIMA.



(a) Solution Length (strlen)

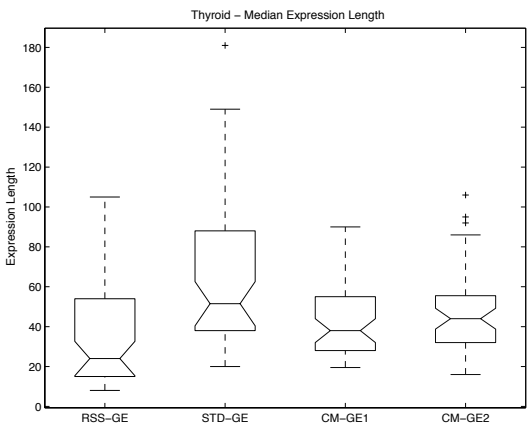


(b) Comparison of Means

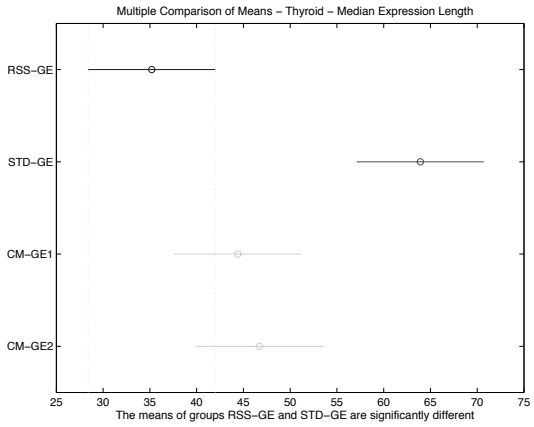


(c) Comparison of Mean Ranks

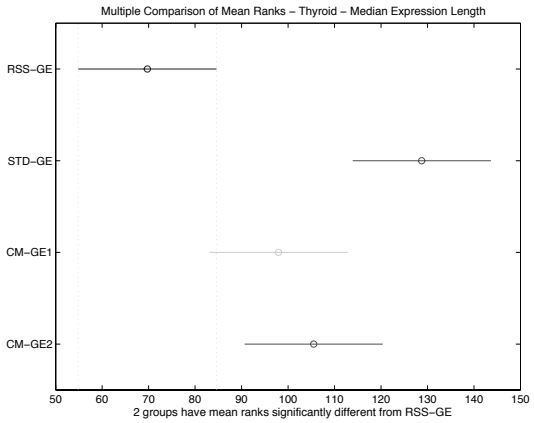
Figure A.41: Direct (GE) comparison of solution length on SHUT.



(a) Solution Length (strlen)

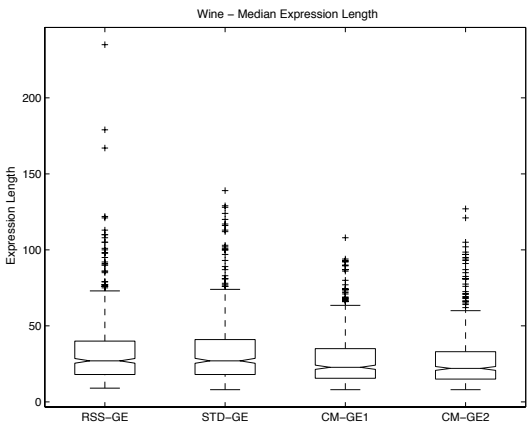


(b) Comparison of Means

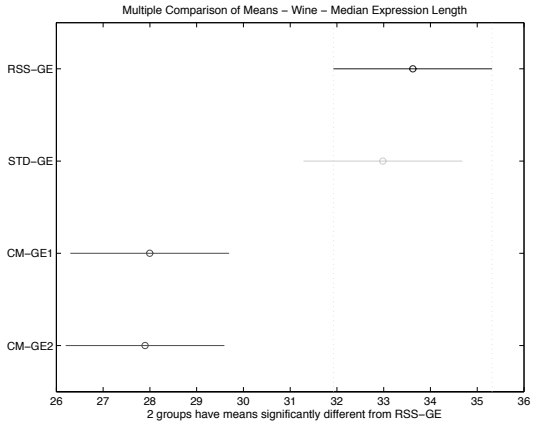


(c) Comparison of Mean Ranks

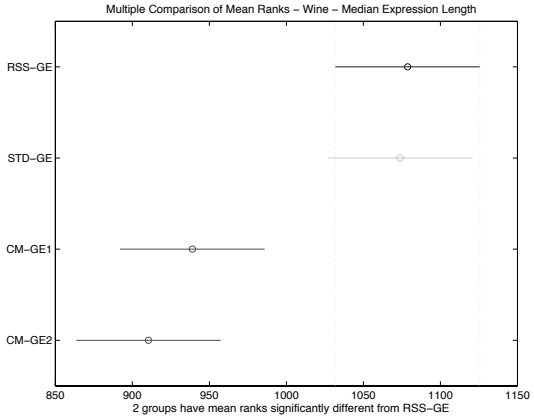
Figure A.42: Direct (GE) comparison of solution length on THYD.



(a) Solution Length (strlen)

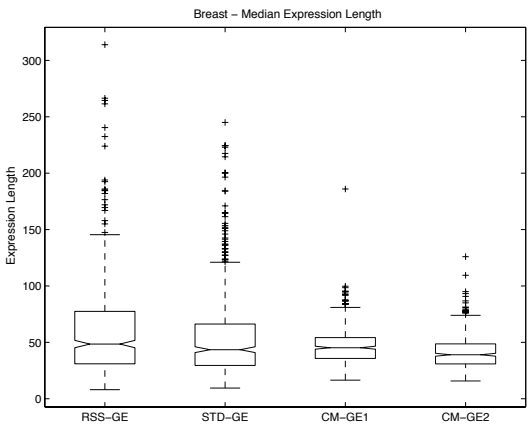


(b) Comparison of Means

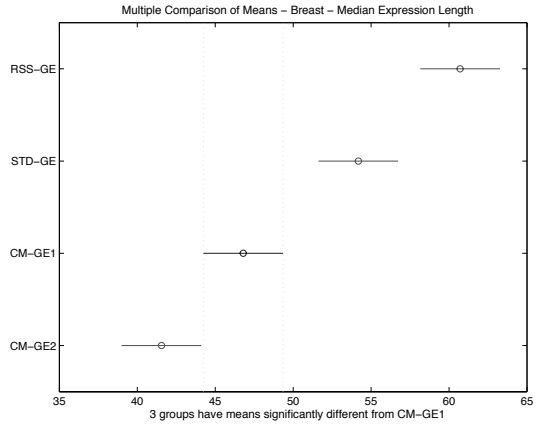


(c) Comparison of Mean Ranks

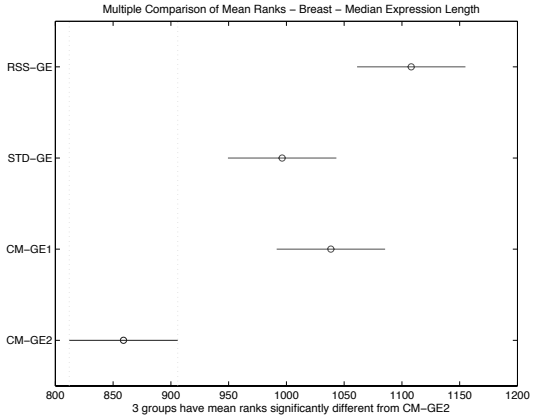
Figure A.43: Direct (GE) comparison of solution length on WINE.



(a) Solution Length (strlen)



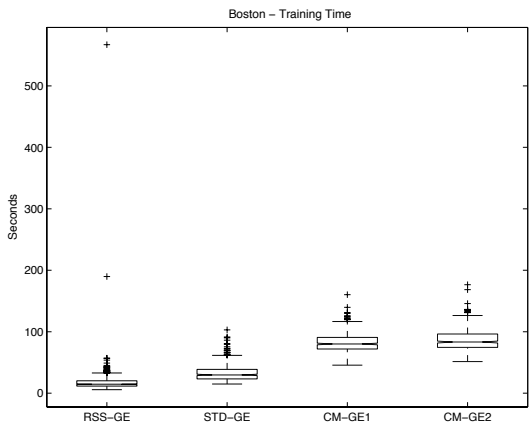
(b) Comparison of Means



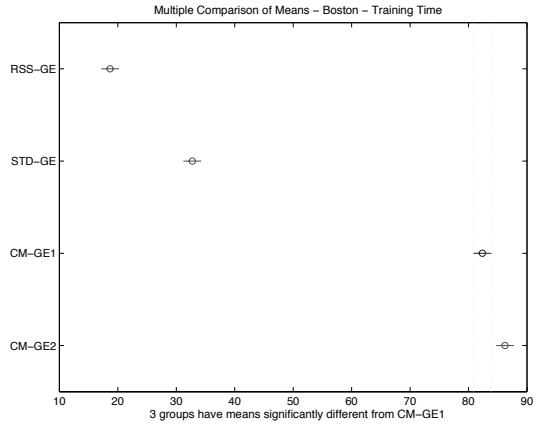
(c) Comparison of Mean Ranks

Figure A.44: Direct (GE) comparison of solution length on WISC.

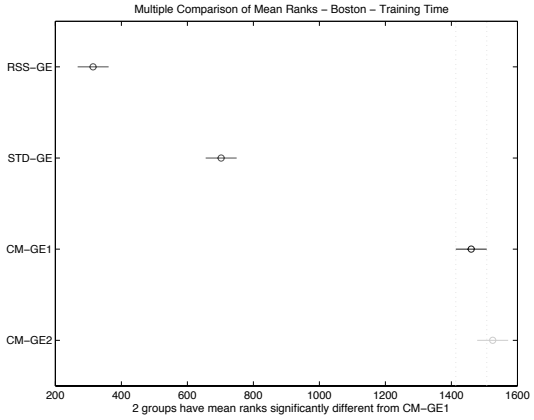
A.2.4 Training Time



(a) Training Time

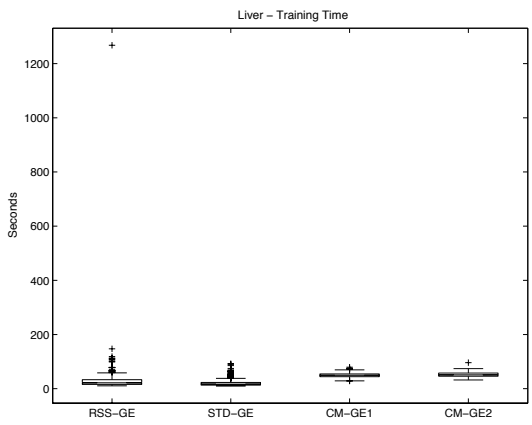


(b) Comparison of Means

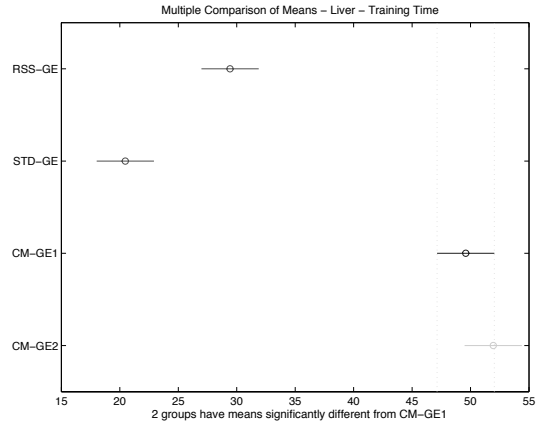


(c) Comparison of Mean Ranks

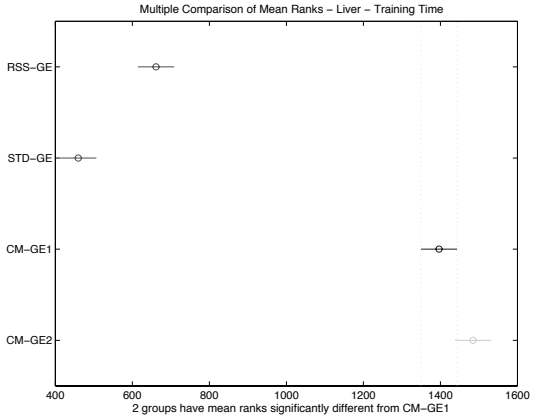
Figure A.45: Direct (GE) comparison of training time on BOST.



(a) Training Time

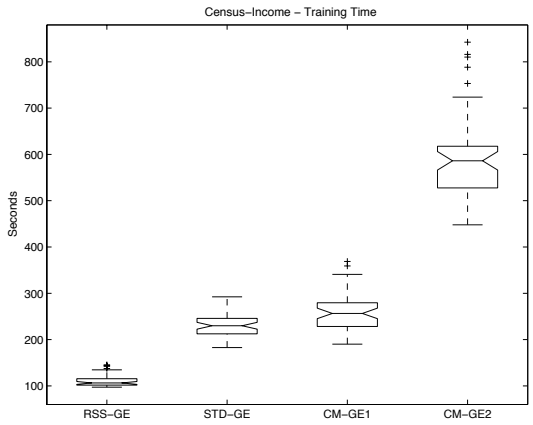


(b) Comparison of Means

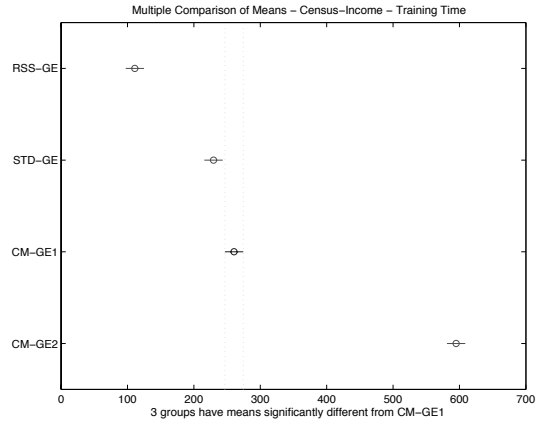


(c) Comparison of Mean Ranks

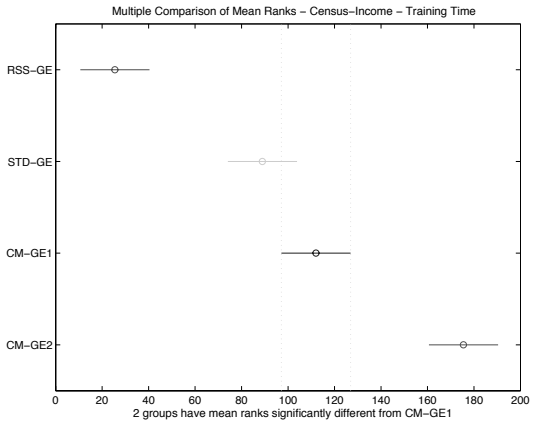
Figure A.46: Direct (GE) comparison of training time on BUPA.



(a) Training Time

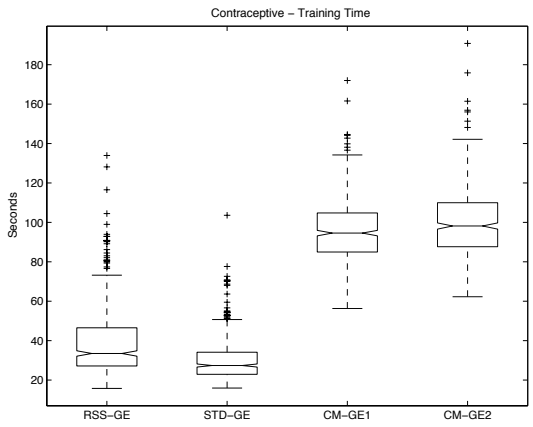


(b) Comparison of Means

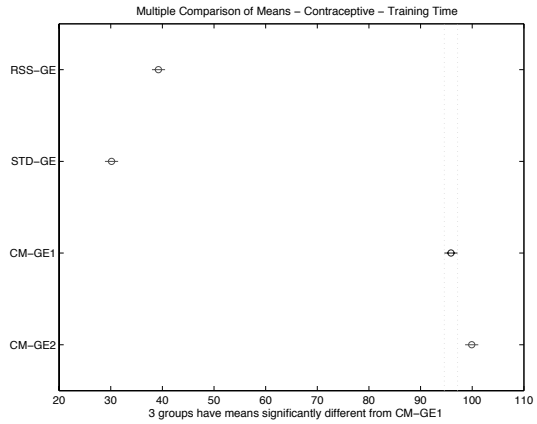


(c) Comparison of Mean Ranks

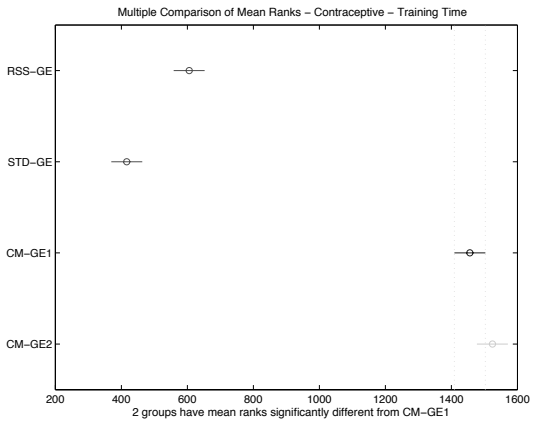
Figure A.47: Direct (GE) comparison of training time on CENS.



(a) Training Time

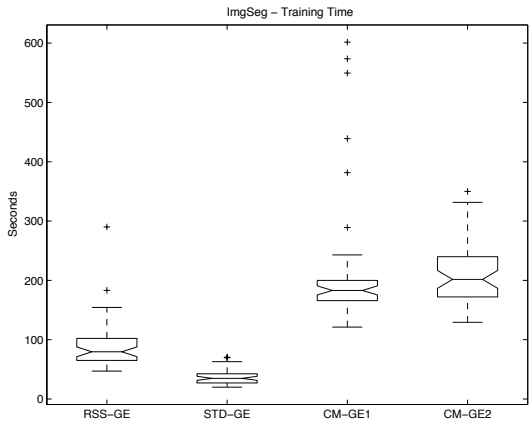


(b) Comparison of Means

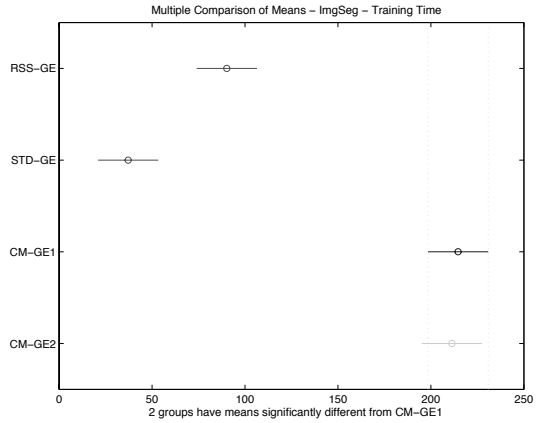


(c) Comparison of Mean Ranks

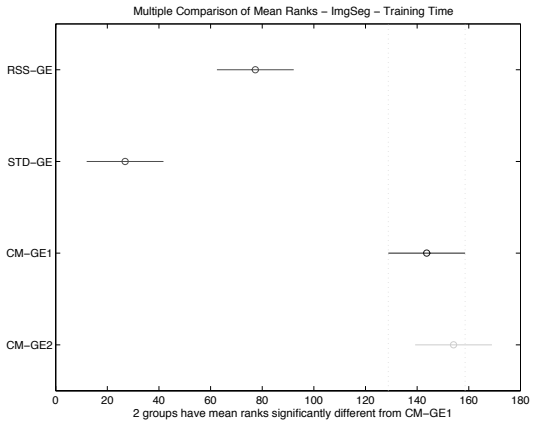
Figure A.48: Direct (GE) comparison of training time on CONT.



(a) Training Time

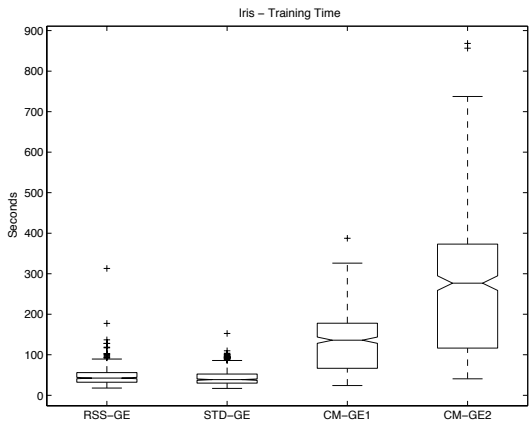


(b) Comparison of Means

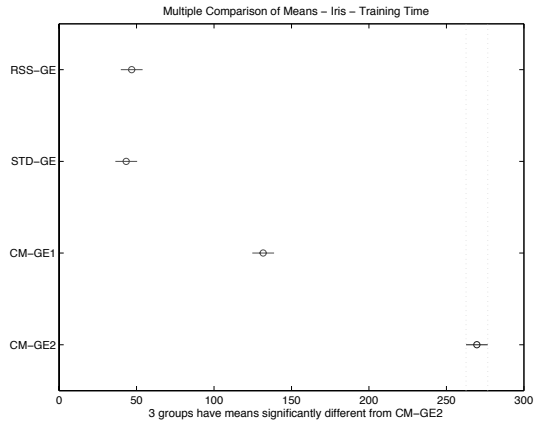


(c) Comparison of Mean Ranks

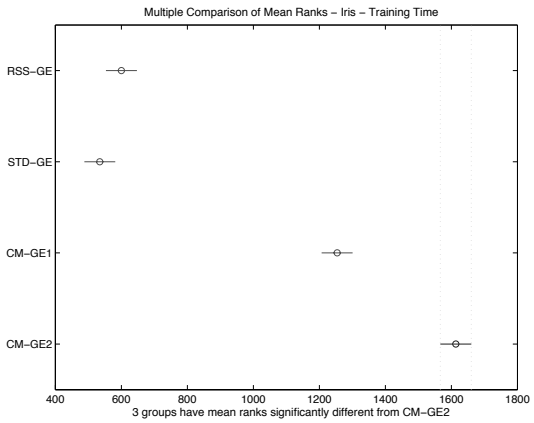
Figure A.49: Direct (GE) comparison of training time on IMAG.



(a) Training Time

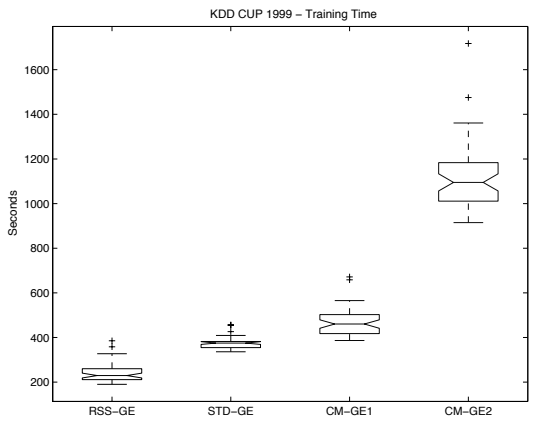


(b) Comparison of Means

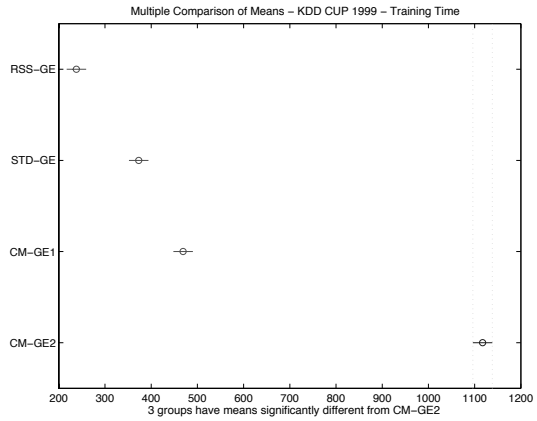


(c) Comparison of Mean Ranks

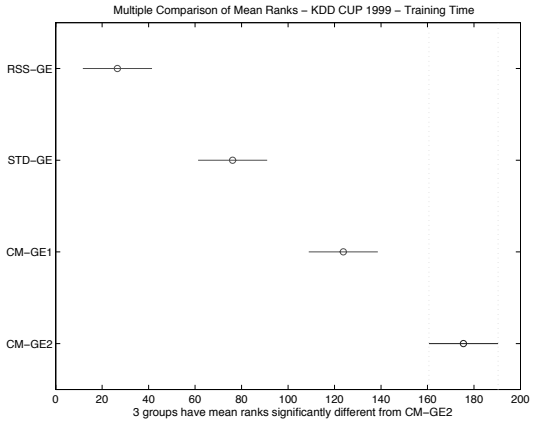
Figure A.50: Direct (GE) comparison of training time on IRIS.



(a) Training Time

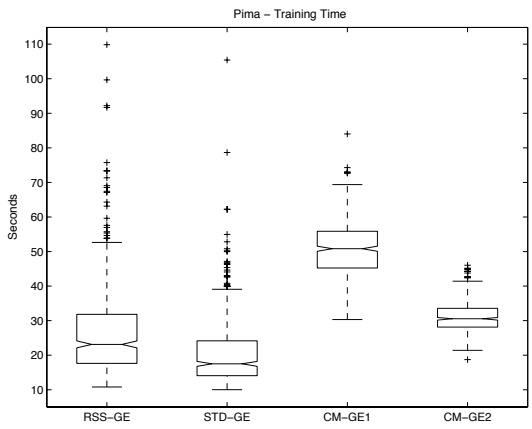


(b) Comparison of Means

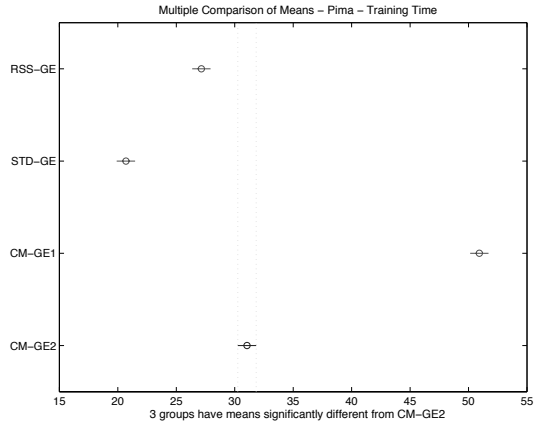


(c) Comparison of Mean Ranks

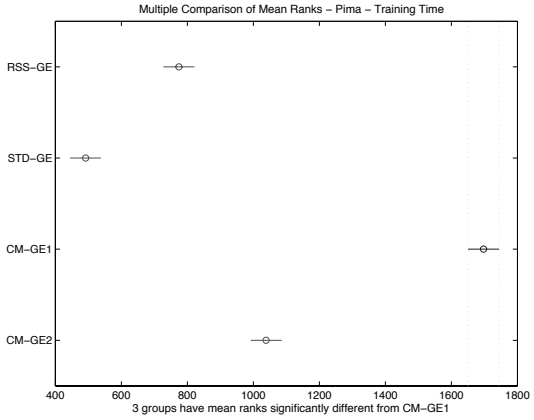
Figure A.51: Direct (GE) comparison of training time on KD99.



(a) Training Time

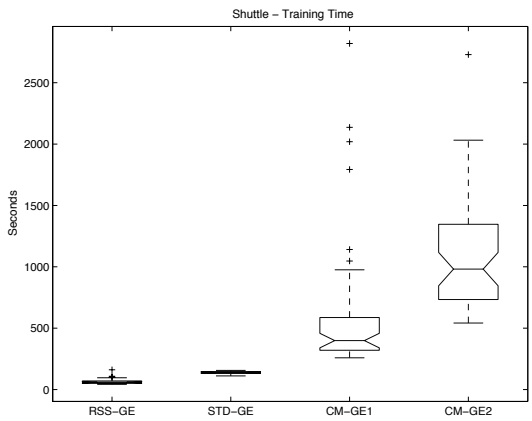


(b) Comparison of Means

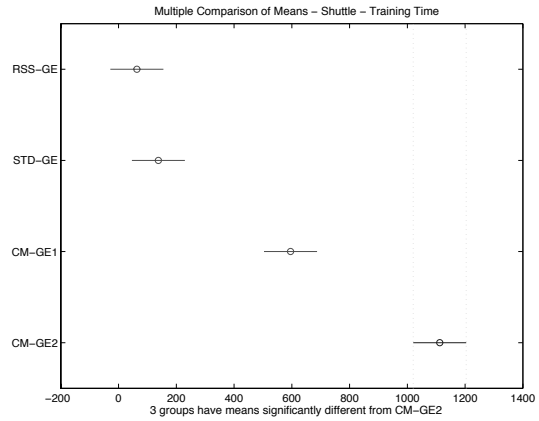


(c) Comparison of Mean Ranks

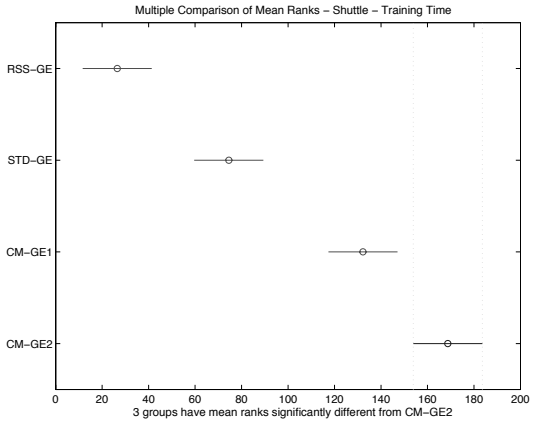
Figure A.52: Direct (GE) comparison of training time on PIMA.



(a) Training Time

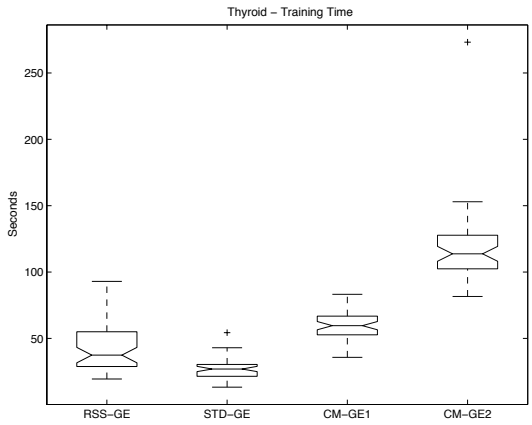


(b) Comparison of Means

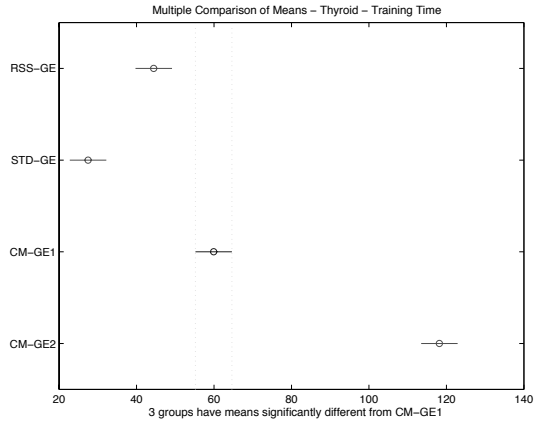


(c) Comparison of Mean Ranks

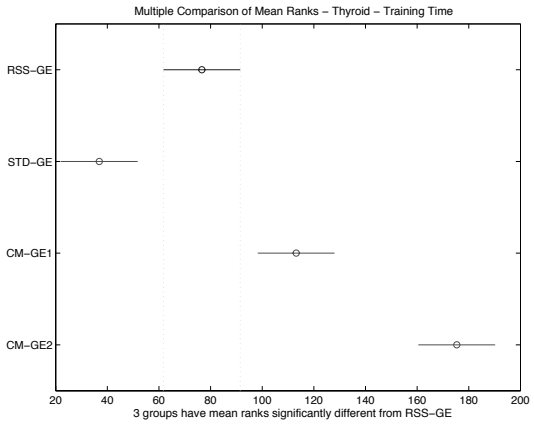
Figure A.53: Direct (GE) comparison of training time on SHUT.



(a) Training Time

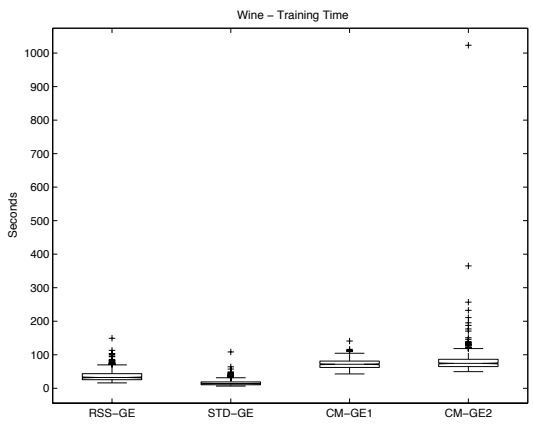


(b) Comparison of Means

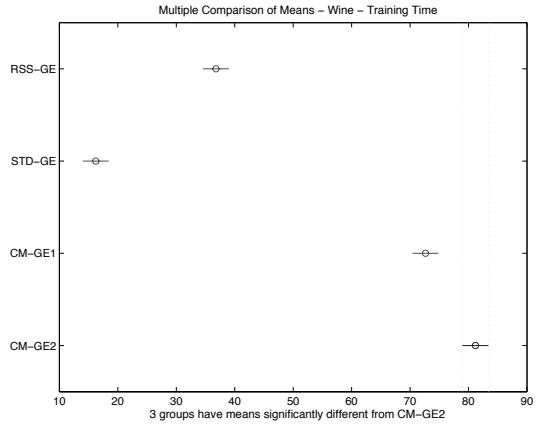


(c) Comparison of Mean Ranks

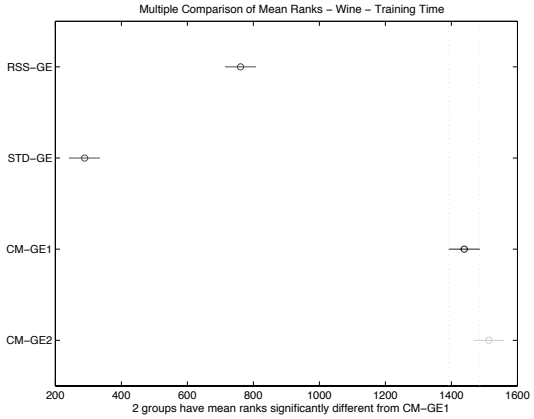
Figure A.54: Direct (GE) comparison of training time on THYD.



(a) Training Time

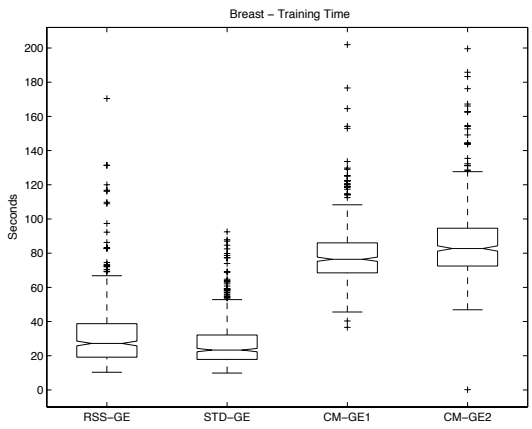


(b) Comparison of Means

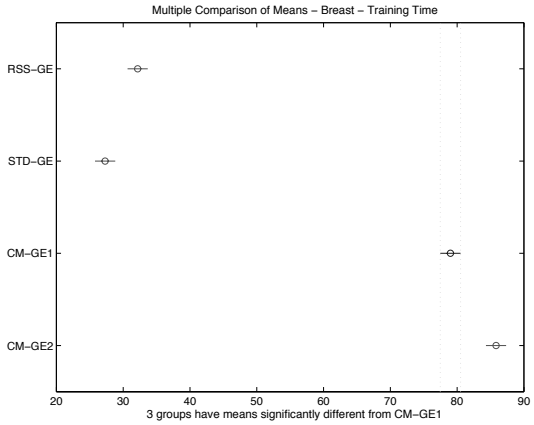


(c) Comparison of Mean Ranks

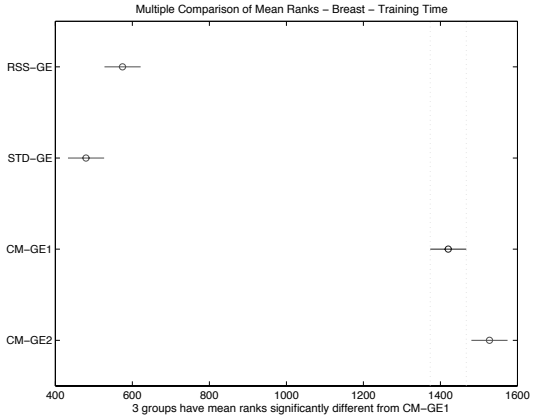
Figure A.55: Direct (GE) comparison of training time on WINE.



(a) Training Time



(b) Comparison of Means



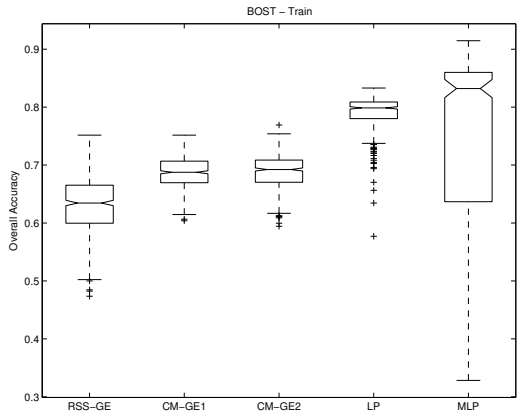
(c) Comparison of Mean Ranks

Figure A.56: Direct (GE) comparison of training time on WISC.

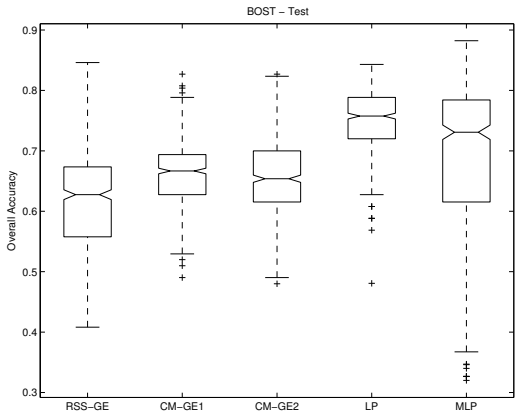
Appendix B

Artificial Neural Network Comparison Plots (E6 - E7)

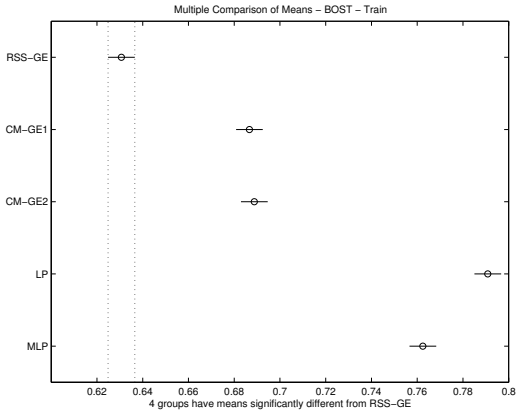
B.1 Overall Accuracy



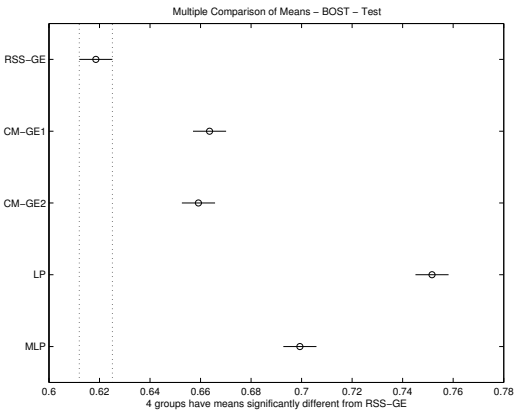
(a) Overall Accuracy (Train)



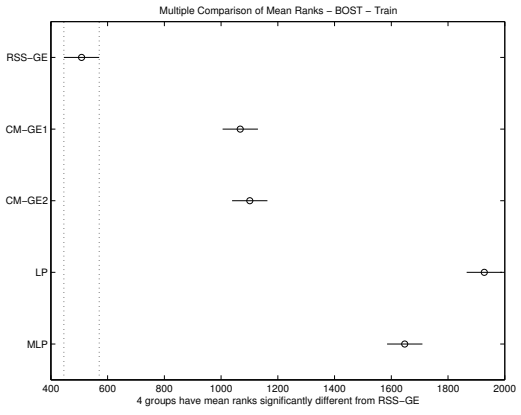
(b) Overall Accuracy (Test)



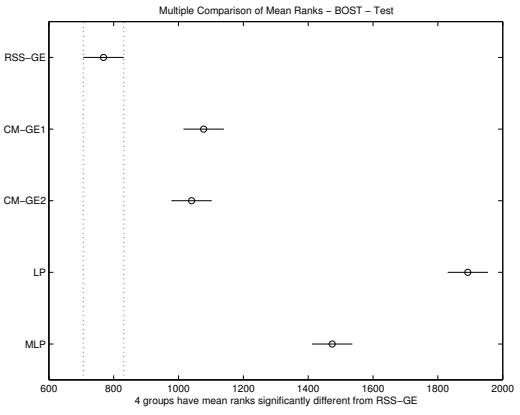
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

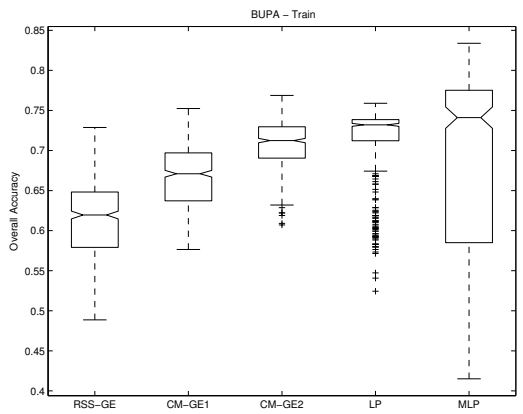


(e) Comparison of Mean Ranks (Train)

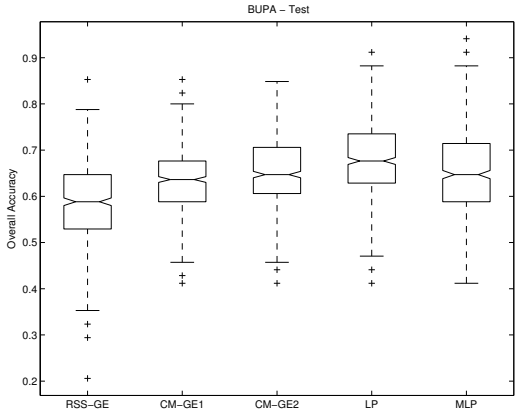


(f) Comparison of Mean Ranks (Test)

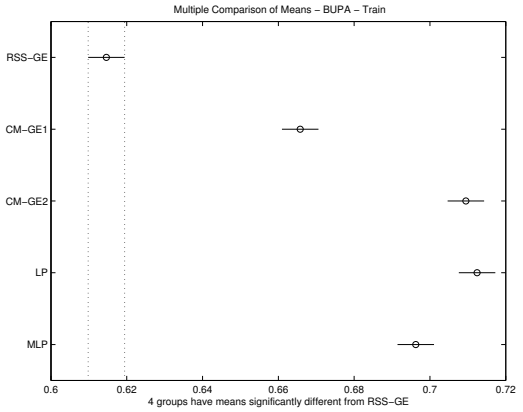
Figure B.1: ANN comparison of BOST Overall Accuracy performance.



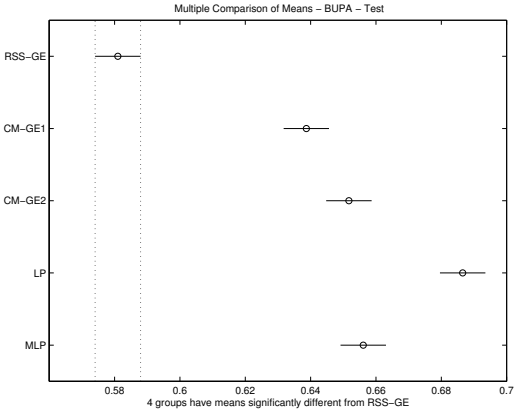
(a) Overall Accuracy (Train)



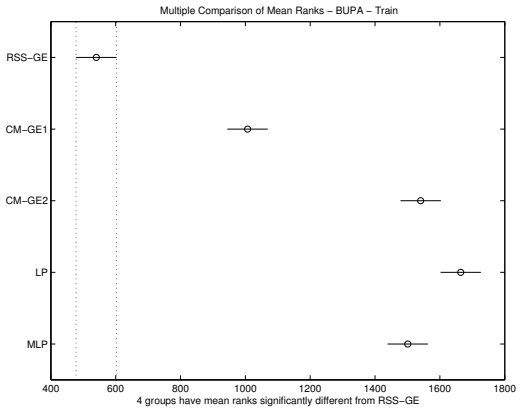
(b) Overall Accuracy (Test)



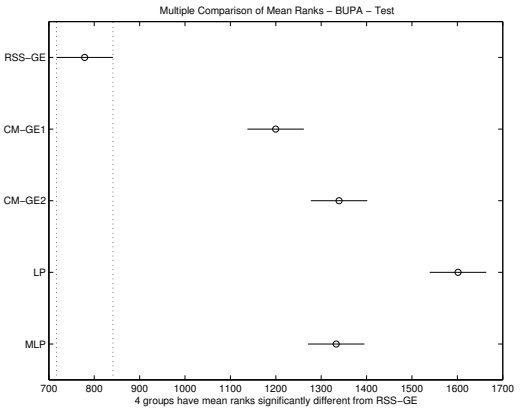
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

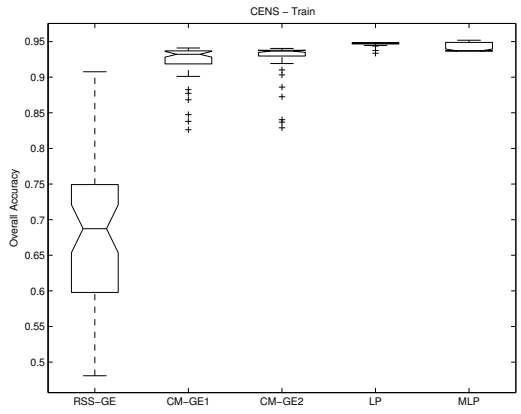


(e) Comparison of Mean Ranks (Train)

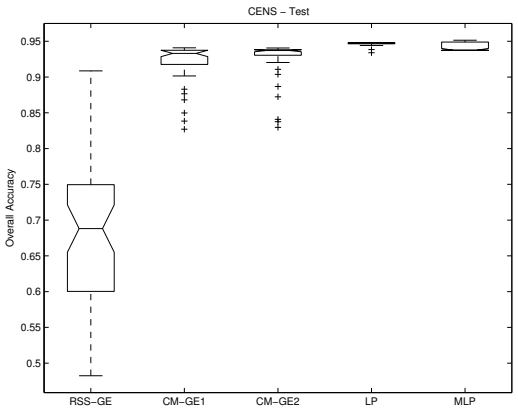


(f) Comparison of Mean Ranks (Test)

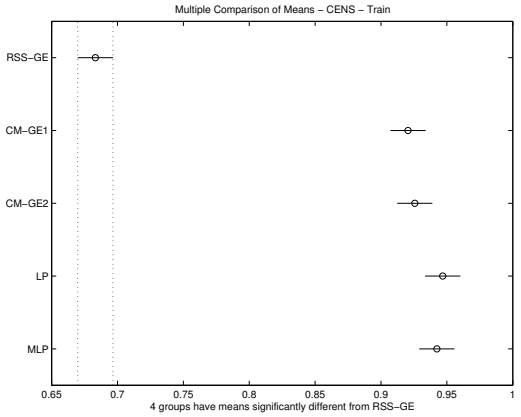
Figure B.2: ANN comparison of BUPA Overall Accuracy performance.



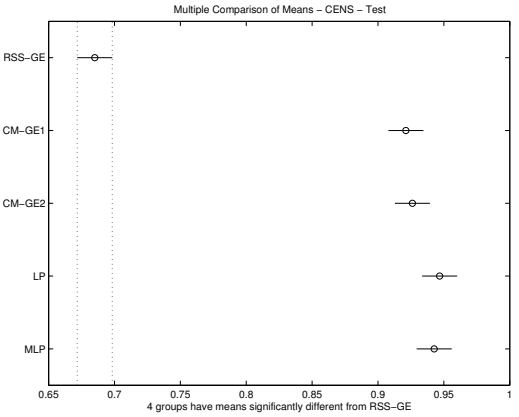
(a) Overall Accuracy (Train)



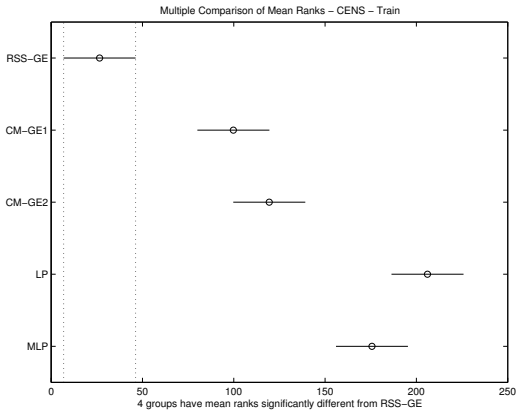
(b) Overall Accuracy (Test)



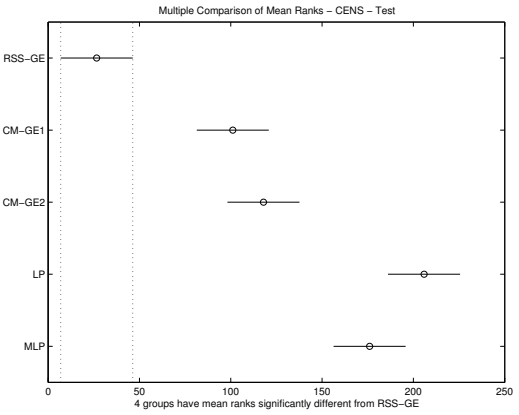
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

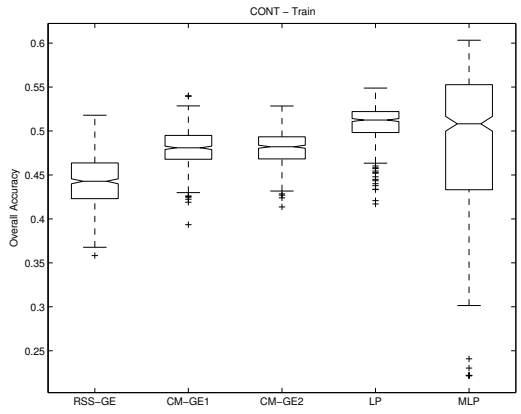


(e) Comparison of Mean Ranks (Train)

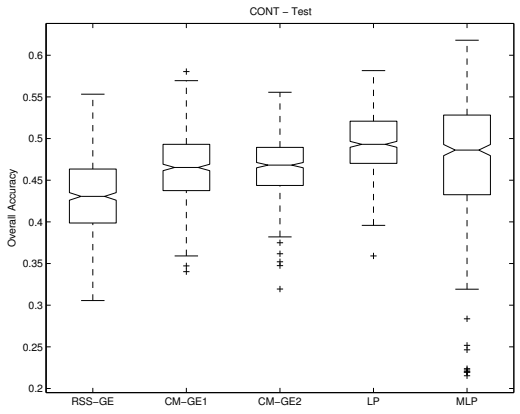


(f) Comparison of Mean Ranks (Test)

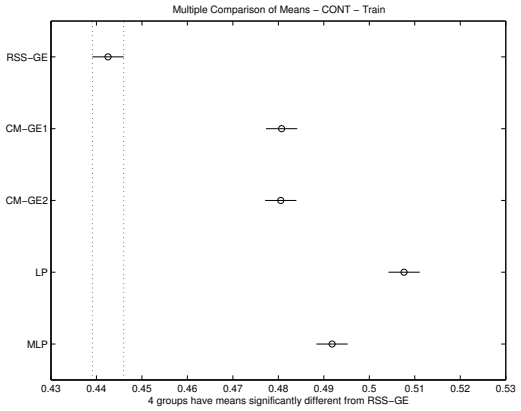
Figure B.3: ANN comparison of CENS Overall Accuracy performance.



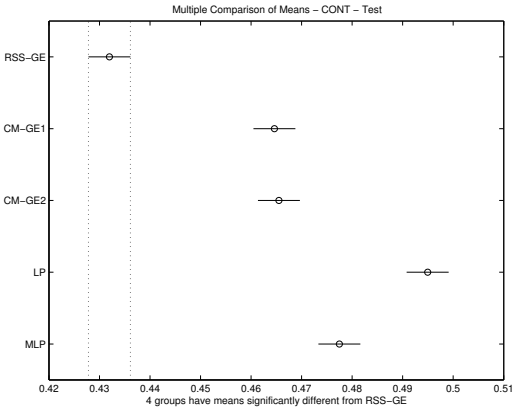
(a) Overall Accuracy (Train)



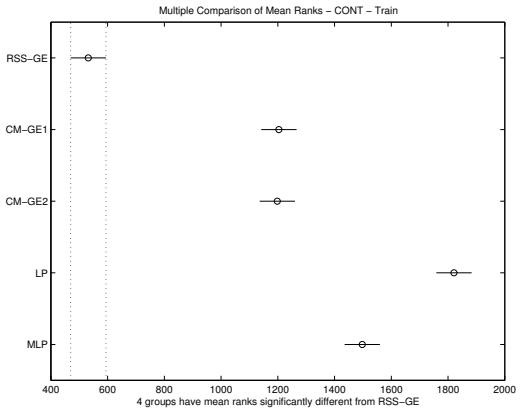
(b) Overall Accuracy (Test)



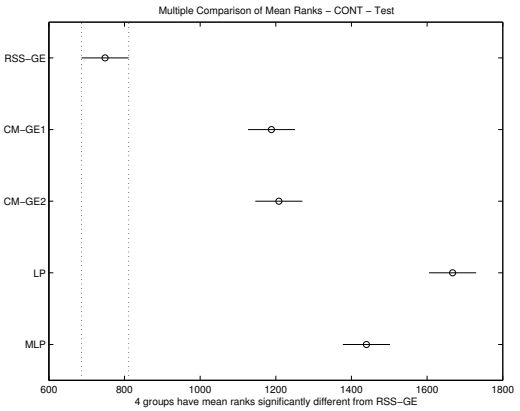
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

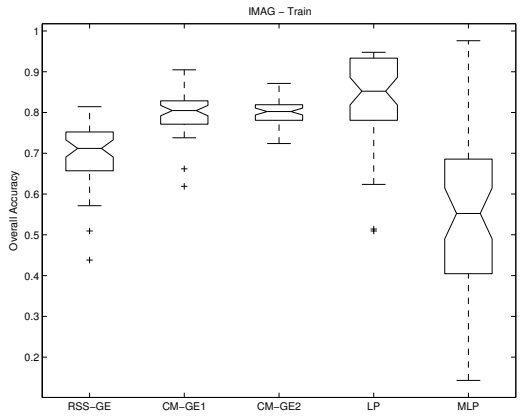


(e) Comparison of Mean Ranks (Train)

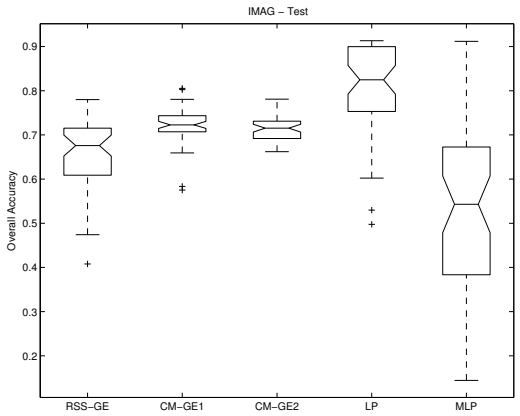


(f) Comparison of Mean Ranks (Test)

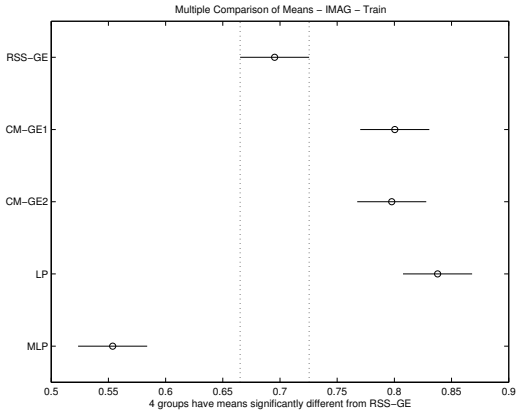
Figure B.4: ANN comparison of CONT Overall Accuracy performance.



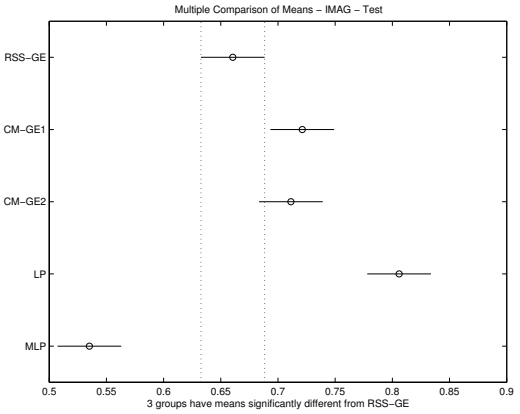
(a) Overall Accuracy (Train)



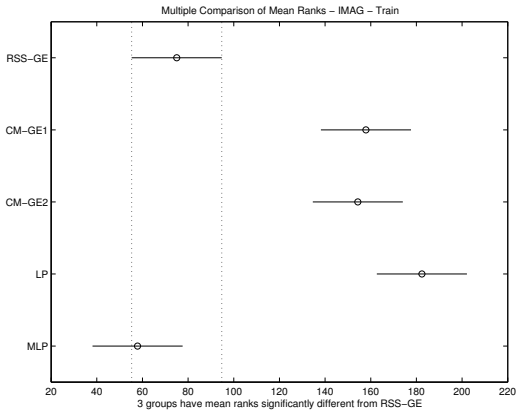
(b) Overall Accuracy (Test)



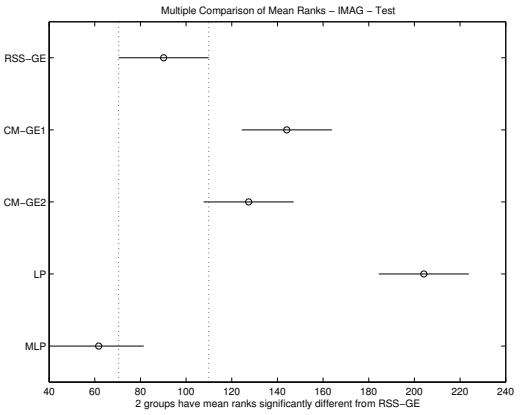
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

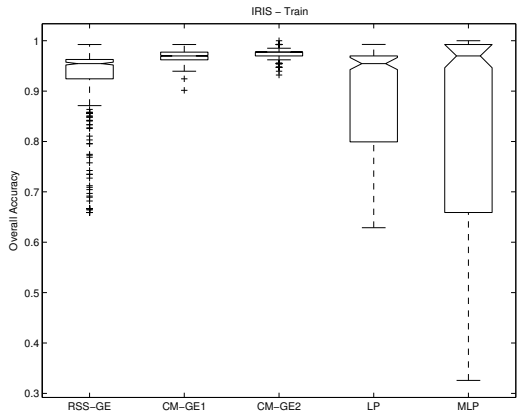


(e) Comparison of Mean Ranks (Train)

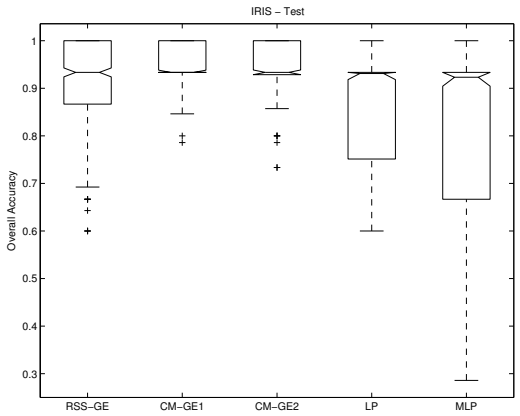


(f) Comparison of Mean Ranks (Test)

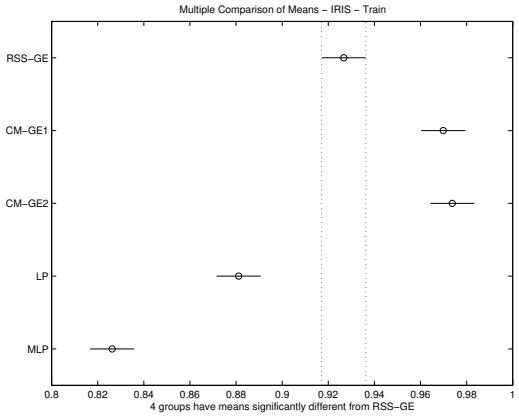
Figure B.5: ANN comparison of IMAG Overall Accuracy performance.



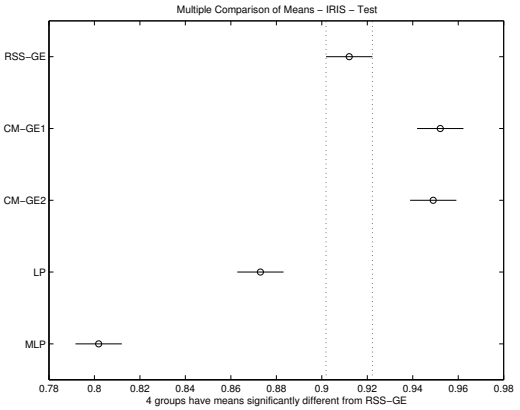
(a) Overall Accuracy (Train)



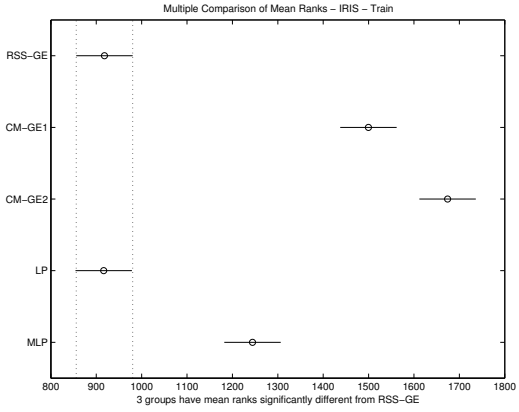
(b) Overall Accuracy (Test)



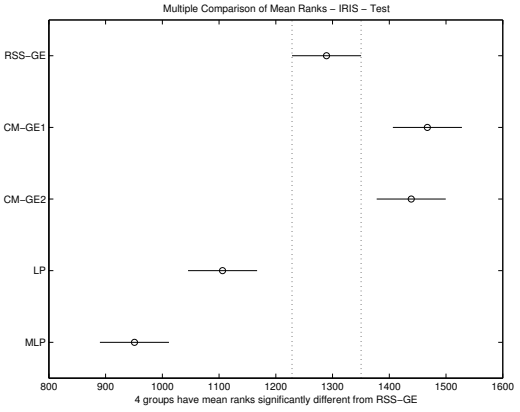
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

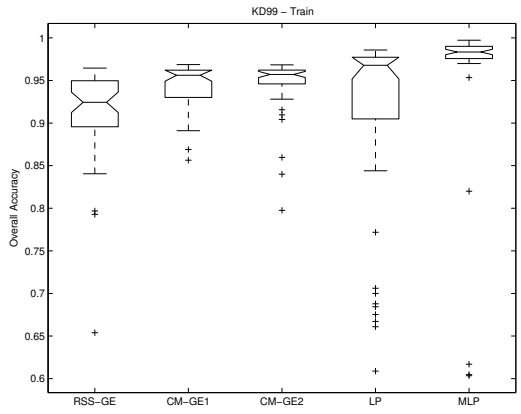


(e) Comparison of Mean Ranks (Train)

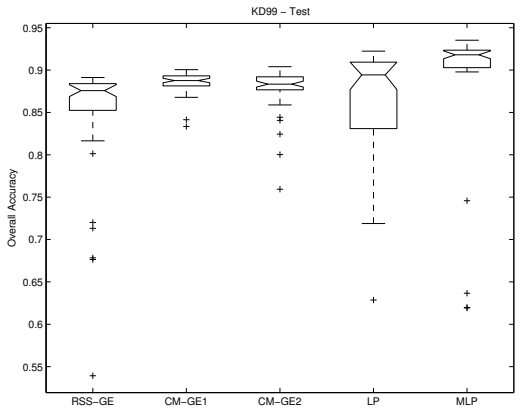


(f) Comparison of Mean Ranks (Test)

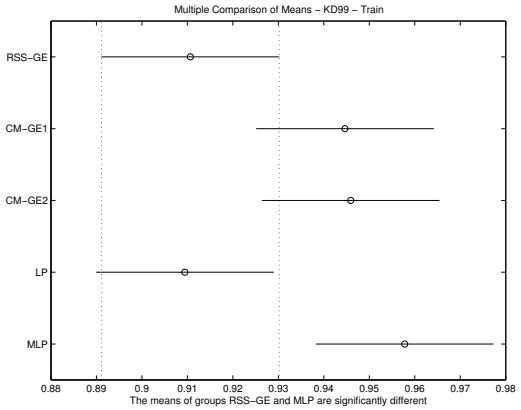
Figure B.6: ANN comparison of IRIS Overall Accuracy performance.



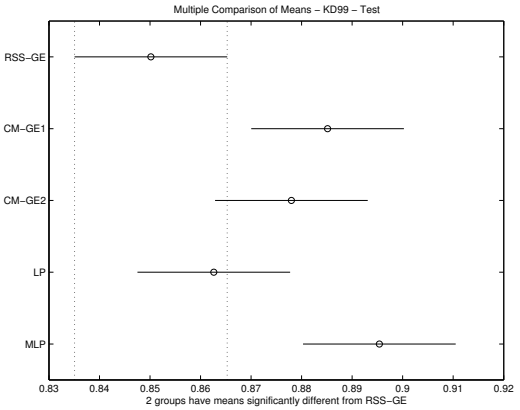
(a) Overall Accuracy (Train)



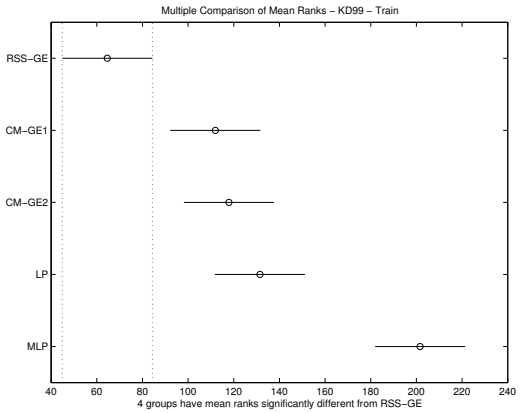
(b) Overall Accuracy (Test)



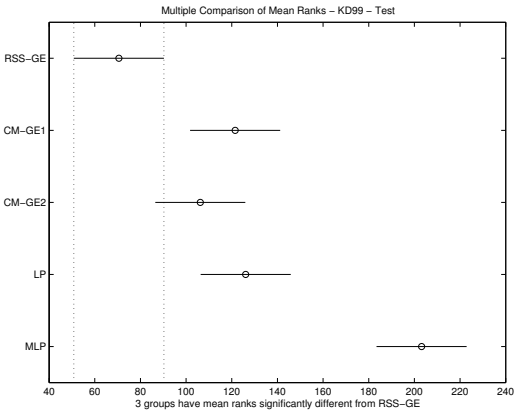
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

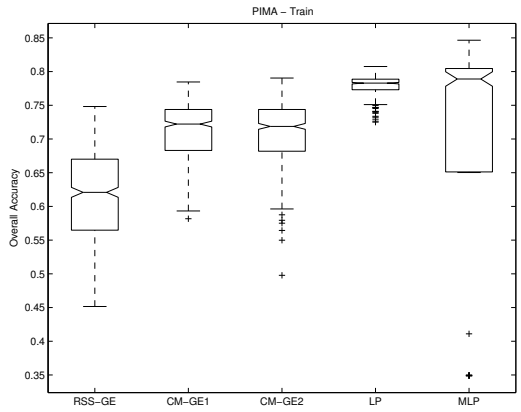


(e) Comparison of Mean Ranks (Train)

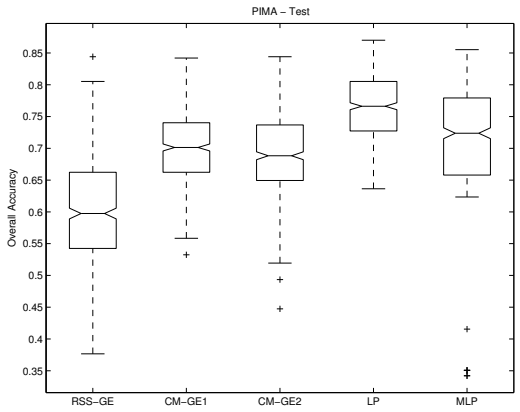


(f) Comparison of Mean Ranks (Test)

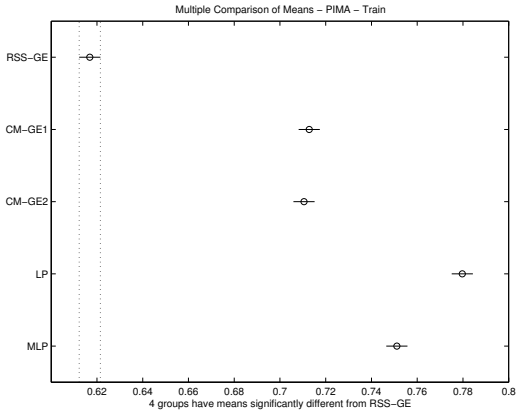
Figure B.7: ANN comparison of KD99 Overall Accuracy performance.



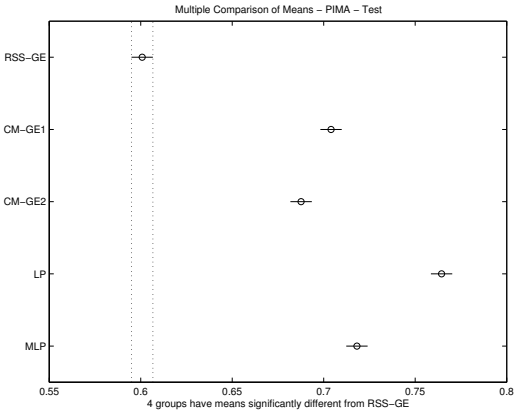
(a) Overall Accuracy (Train)



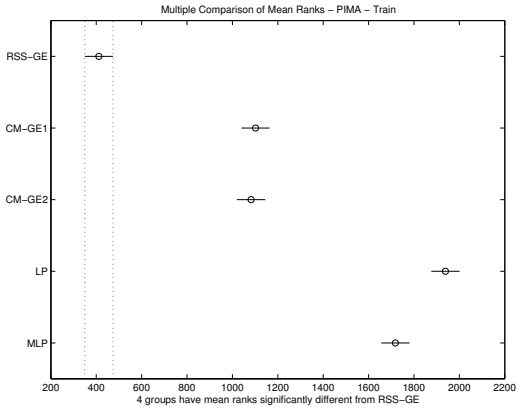
(b) Overall Accuracy (Test)



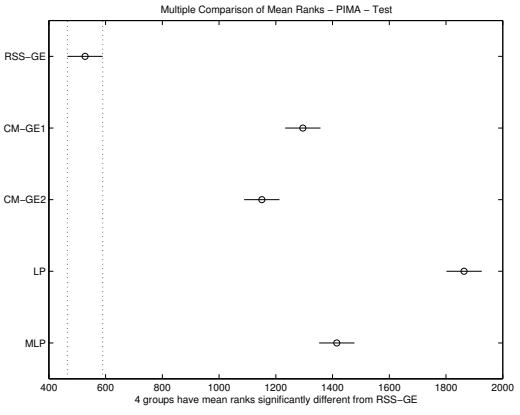
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

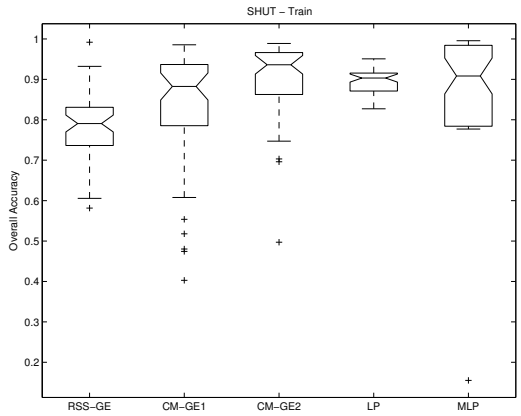


(e) Comparison of Mean Ranks (Train)

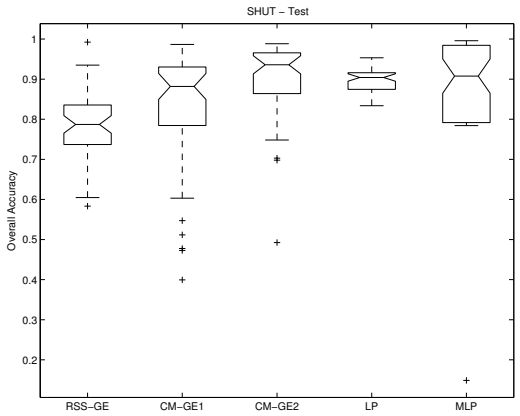


(f) Comparison of Mean Ranks (Test)

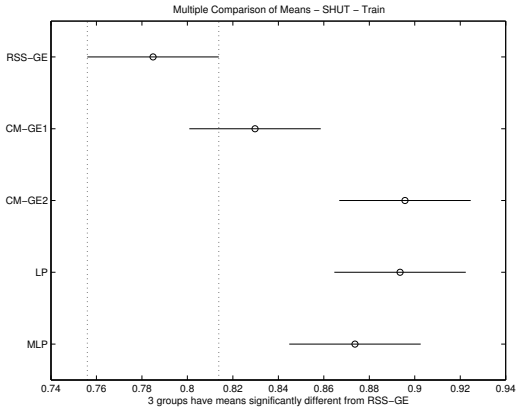
Figure B.8: ANN comparison of PIMA Overall Accuracy performance.



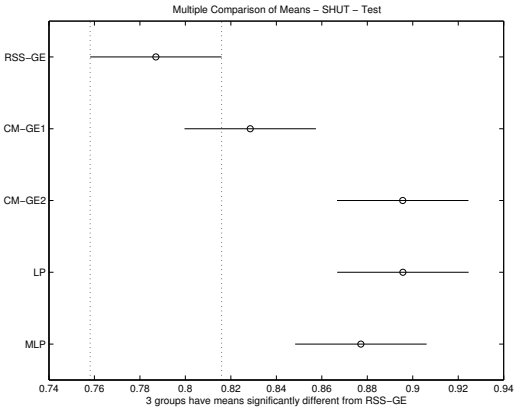
(a) Overall Accuracy (Train)



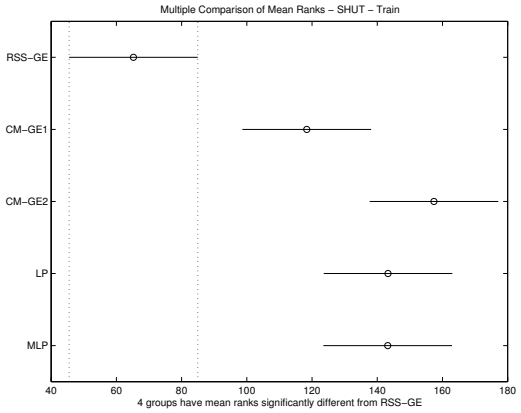
(b) Overall Accuracy (Test)



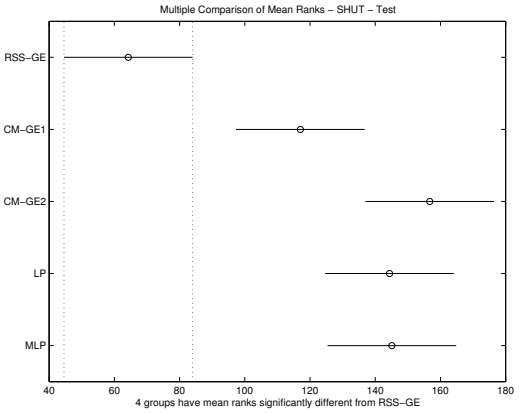
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

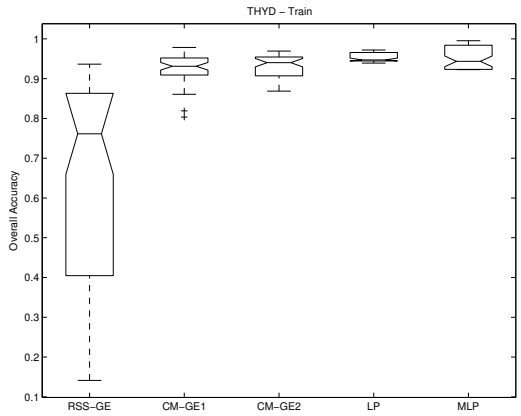


(e) Comparison of Mean Ranks (Train)

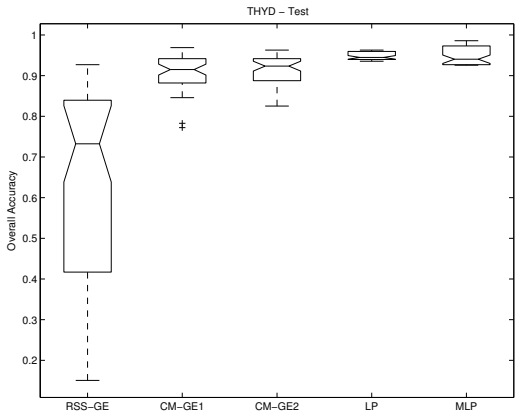


(f) Comparison of Mean Ranks (Test)

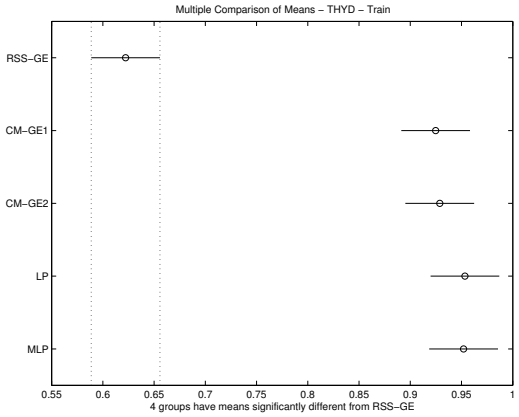
Figure B.9: ANN comparison of SHUT Overall Accuracy performance.



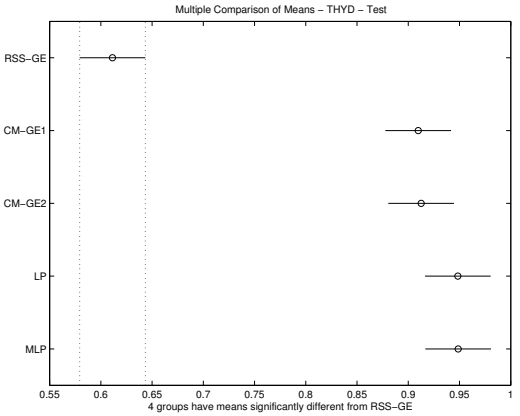
(a) Overall Accuracy (Train)



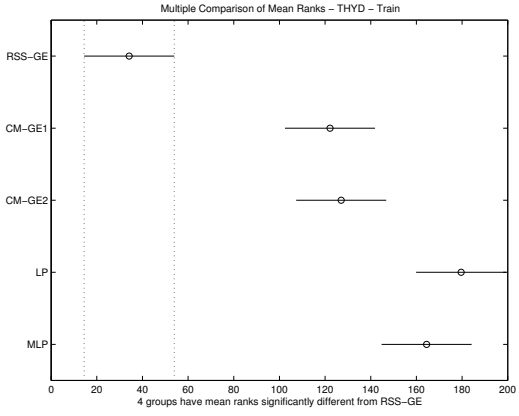
(b) Overall Accuracy (Test)



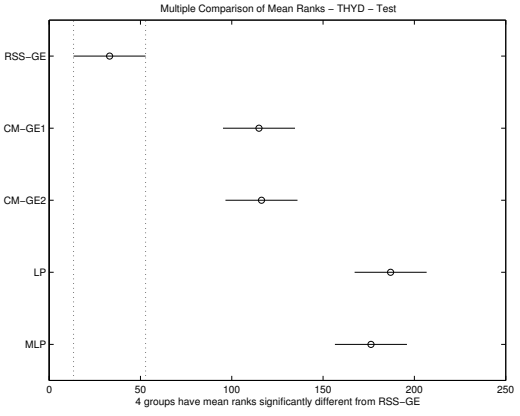
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

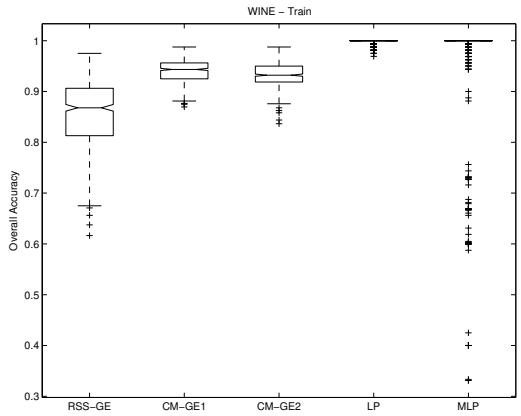


(e) Comparison of Mean Ranks (Train)

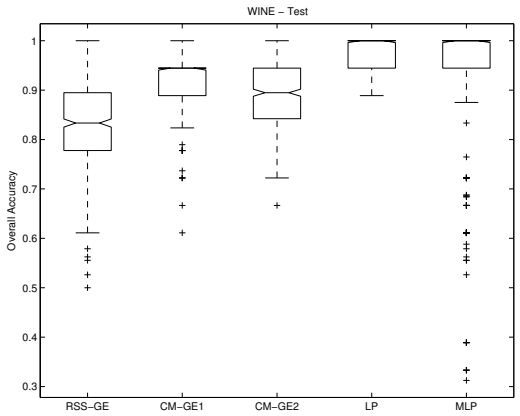


(f) Comparison of Mean Ranks (Test)

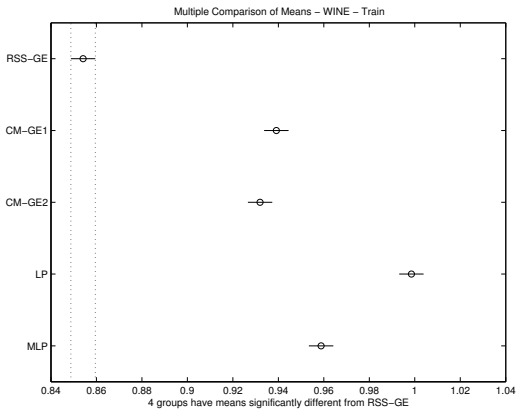
Figure B.10: ANN comparison of THYD Overall Accuracy performance.



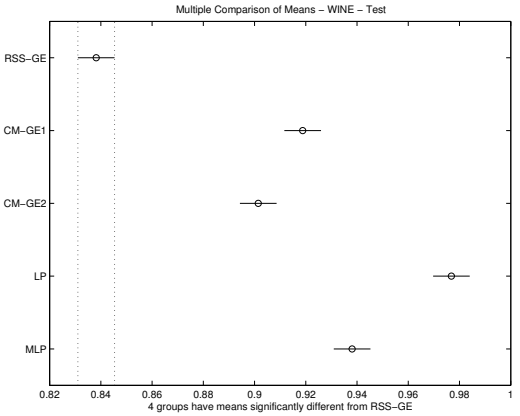
(a) Overall Accuracy (Train)



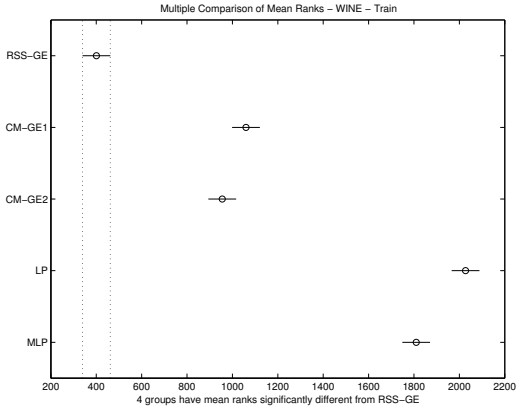
(b) Overall Accuracy (Test)



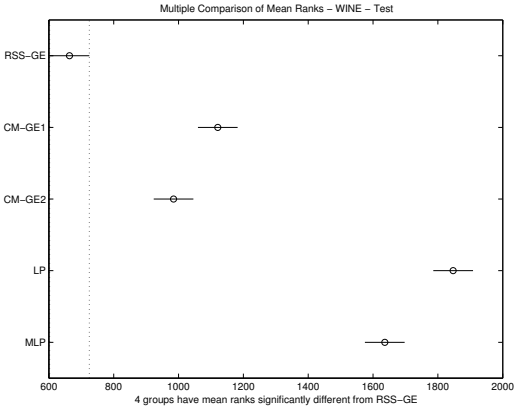
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

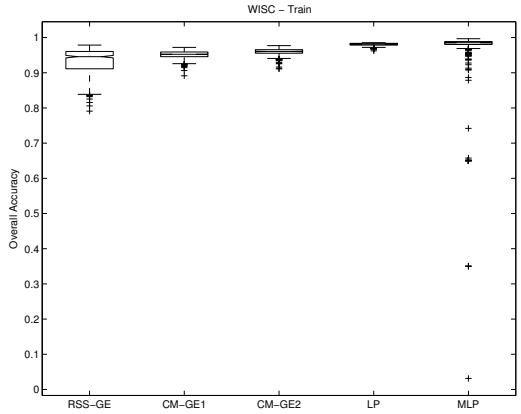


(e) Comparison of Mean Ranks (Train)

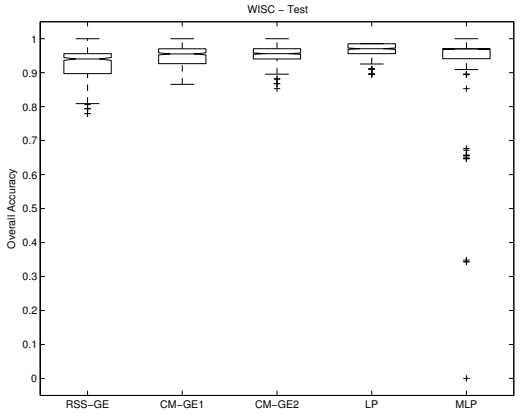


(f) Comparison of Mean Ranks (Test)

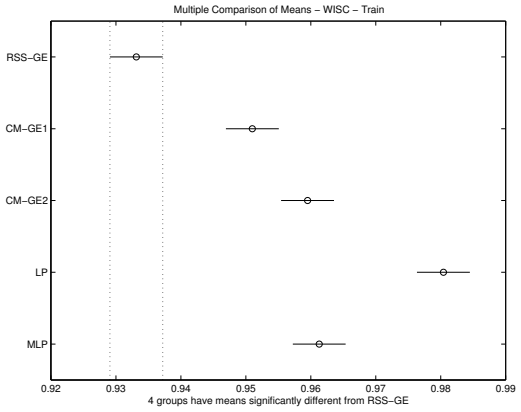
Figure B.11: ANN comparison of WINE Overall Accuracy performance.



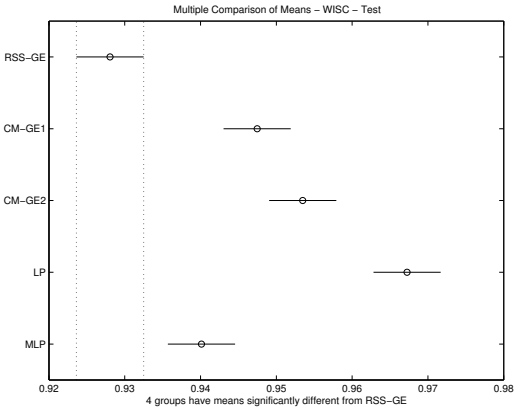
(a) Overall Accuracy (Train)



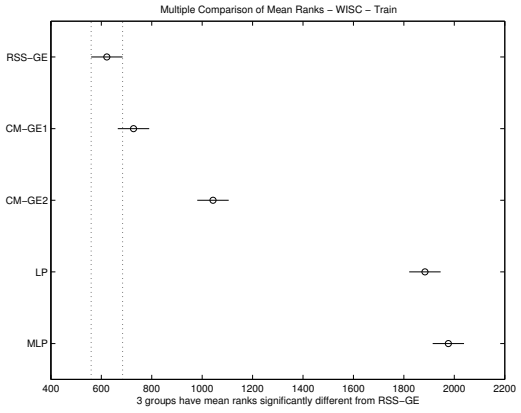
(b) Overall Accuracy (Test)



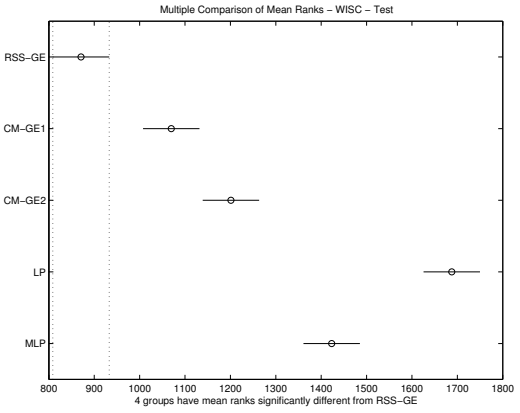
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



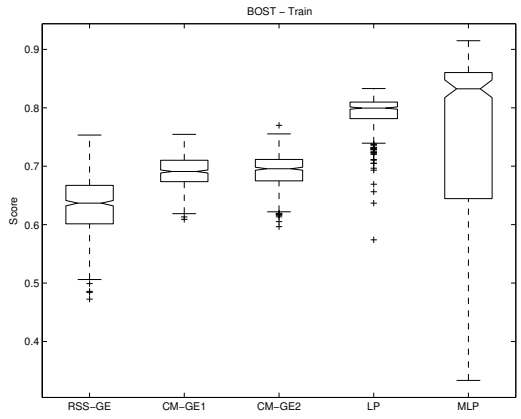
(e) Comparison of Mean Ranks (Train)



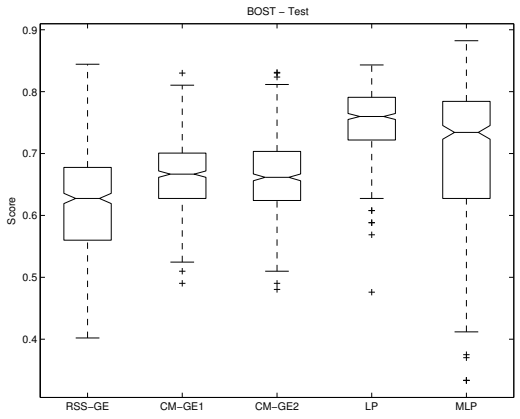
(f) Comparison of Mean Ranks (Test)

Figure B.12: ANN comparison of WISC Overall Accuracy performance.

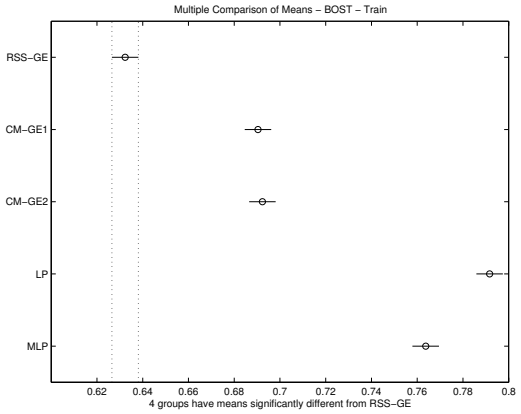
B.2 Score



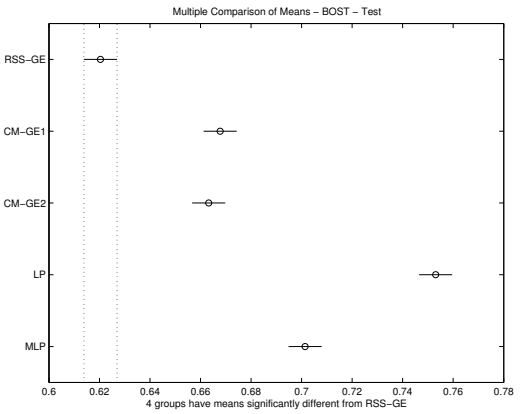
(a) Score (Train)



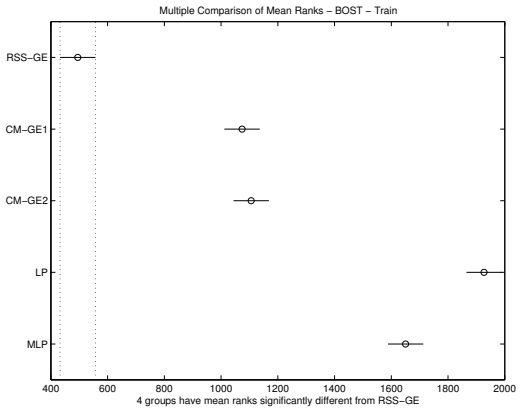
(b) Score (Test)



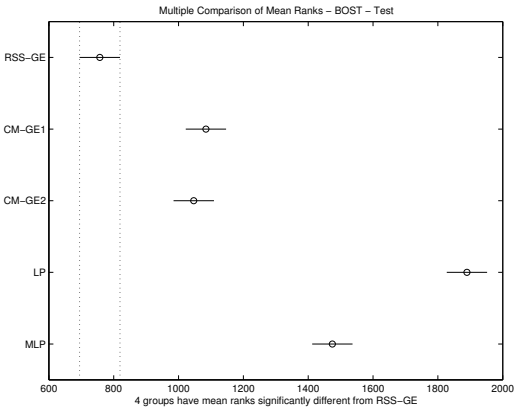
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

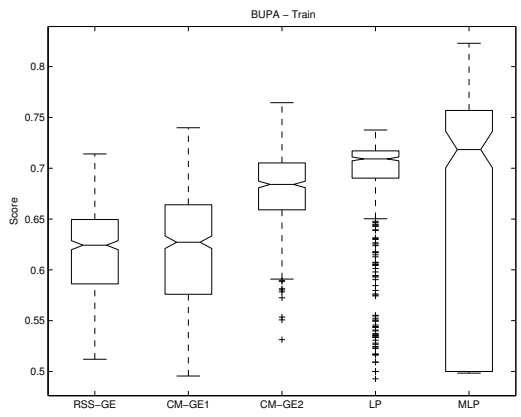


(e) Comparison of Mean Ranks (Train)

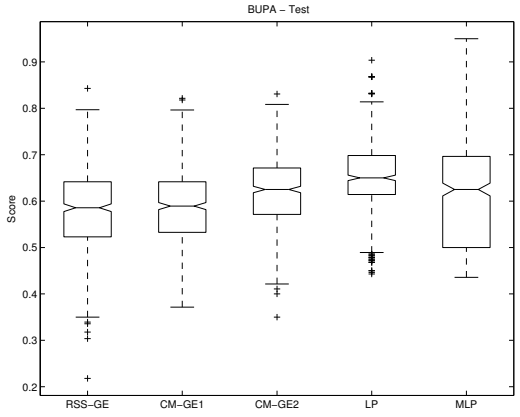


(f) Comparison of Mean Ranks (Test)

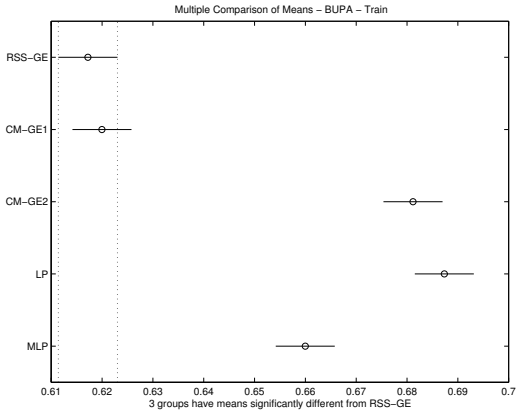
Figure B.13: ANN comparison of BOST Score performance.



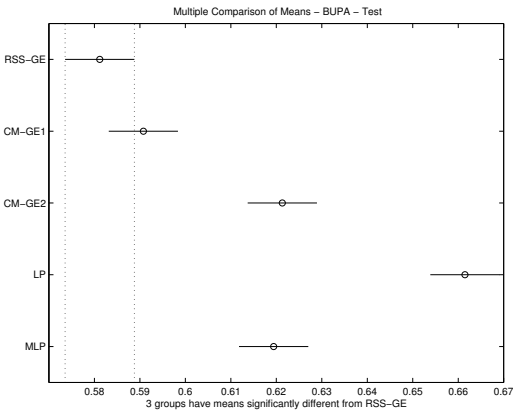
(a) Score (Train)



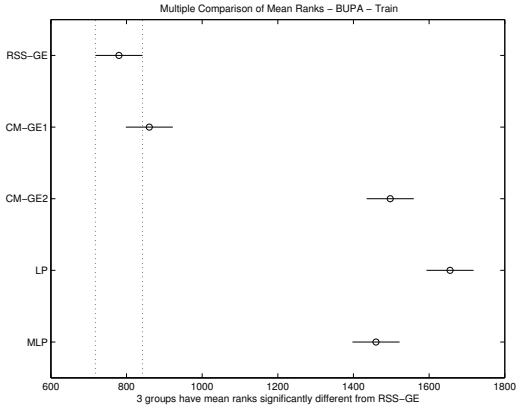
(b) Score (Test)



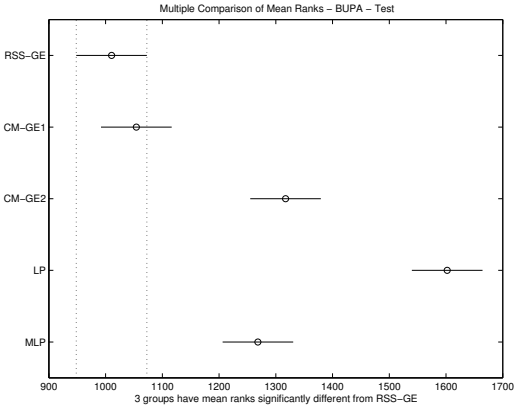
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

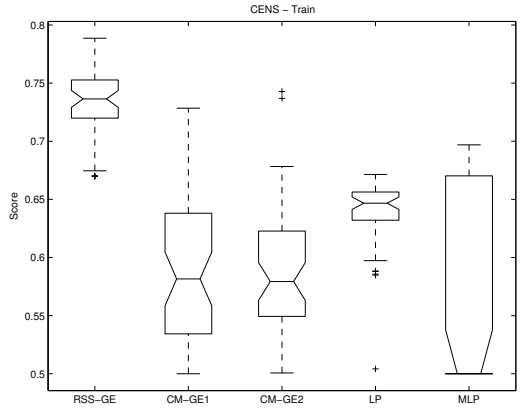


(e) Comparison of Mean Ranks (Train)

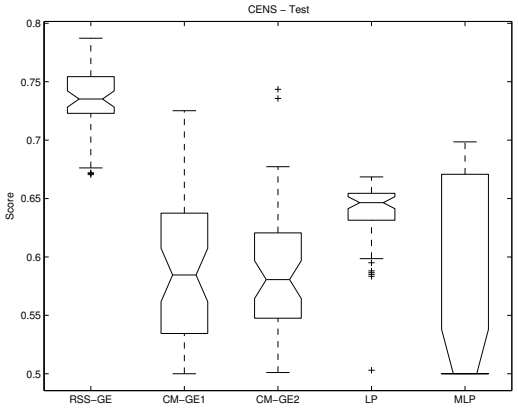


(f) Comparison of Mean Ranks (Test)

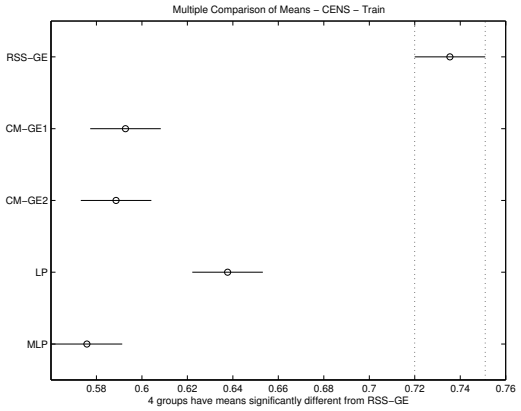
Figure B.14: ANN comparison of BUPA Score performance.



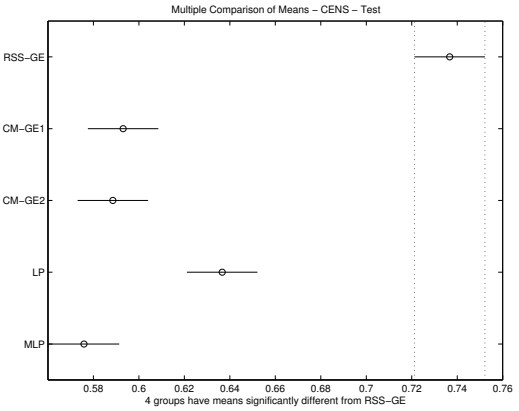
(a) Score (Train)



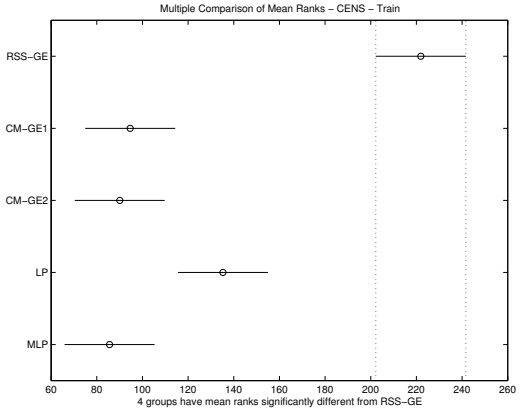
(b) Score (Test)



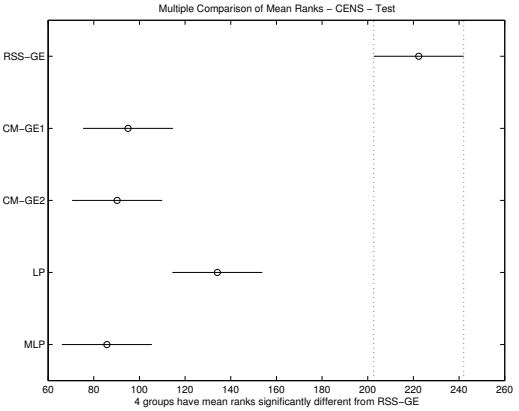
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

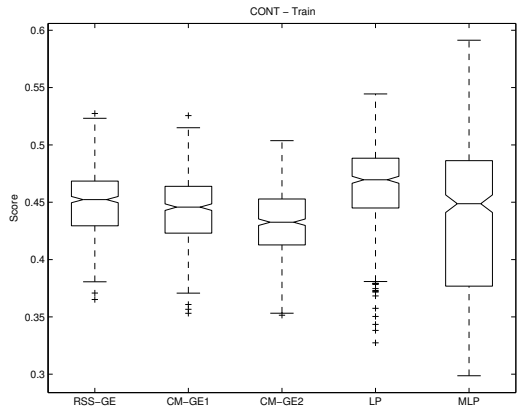


(e) Comparison of Mean Ranks (Train)

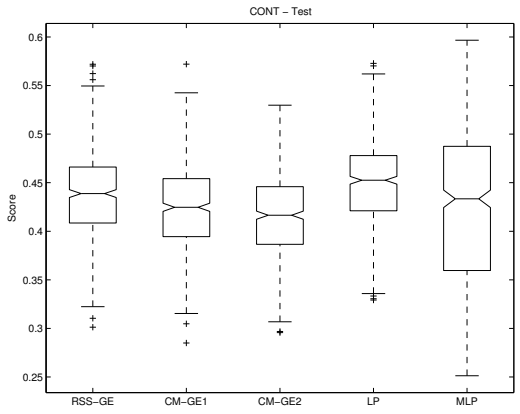


(f) Comparison of Mean Ranks (Test)

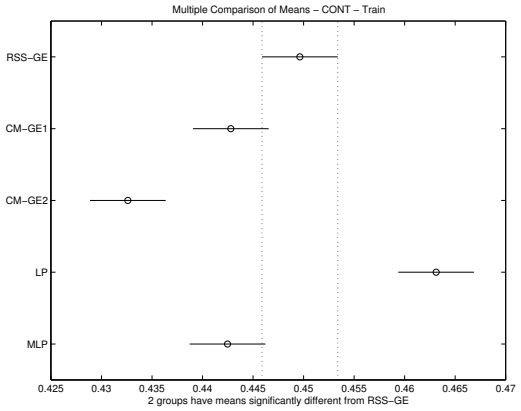
Figure B.15: ANN comparison of CENS Score performance.



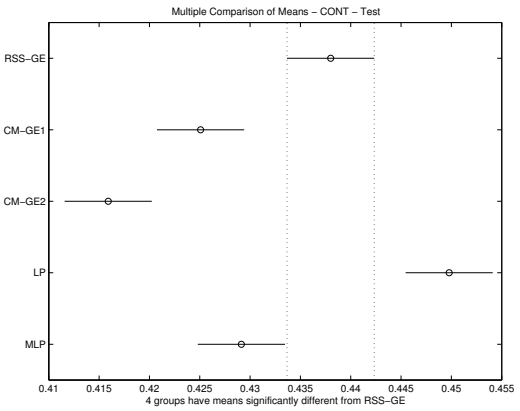
(a) Score (Train)



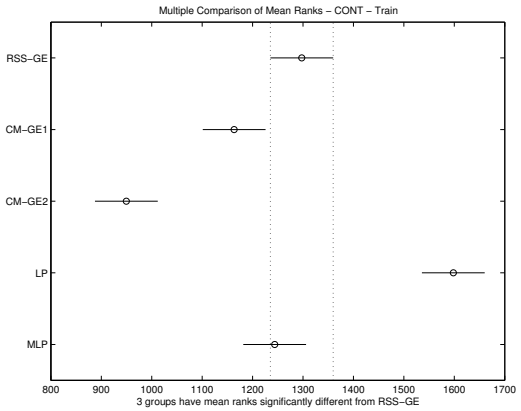
(b) Score (Test)



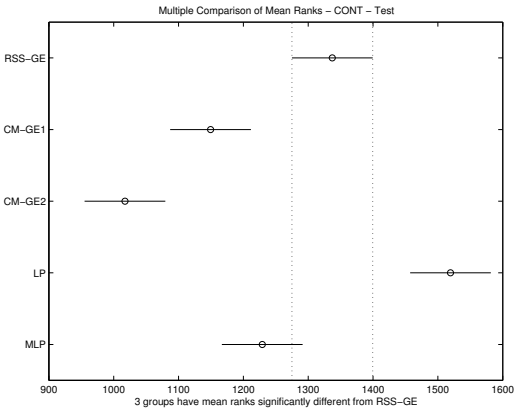
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

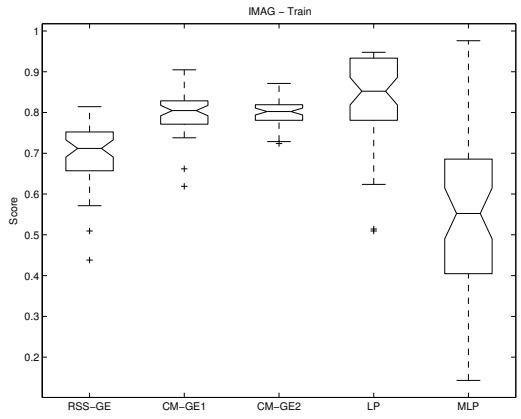


(e) Comparison of Mean Ranks (Train)

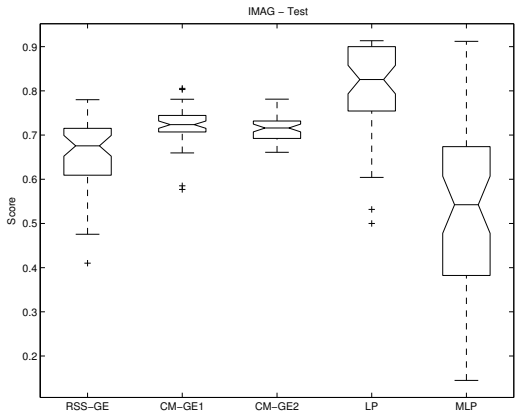


(f) Comparison of Mean Ranks (Test)

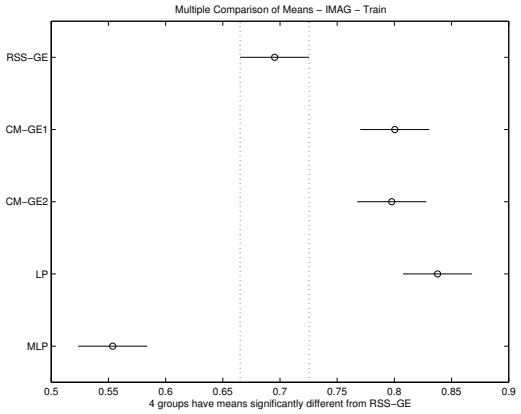
Figure B.16: ANN comparison of CONT Score performance.



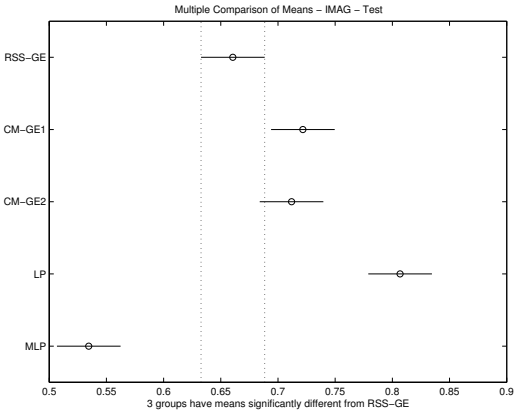
(a) Score (Train)



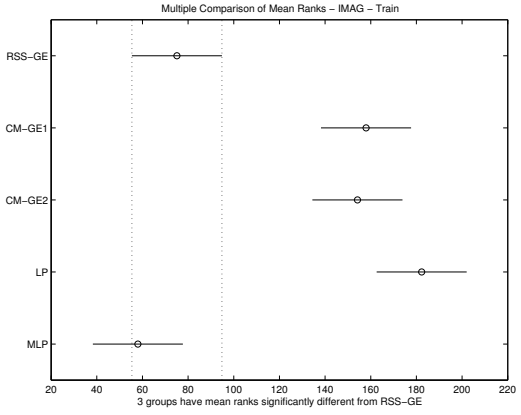
(b) Score (Test)



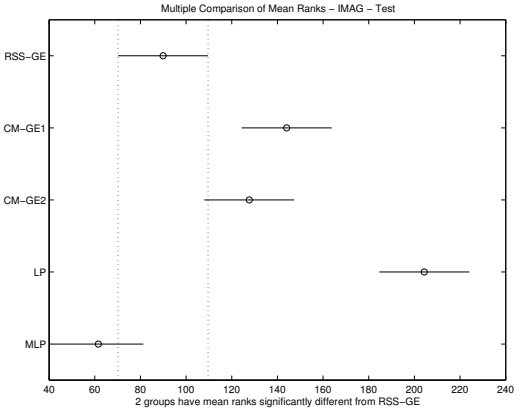
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

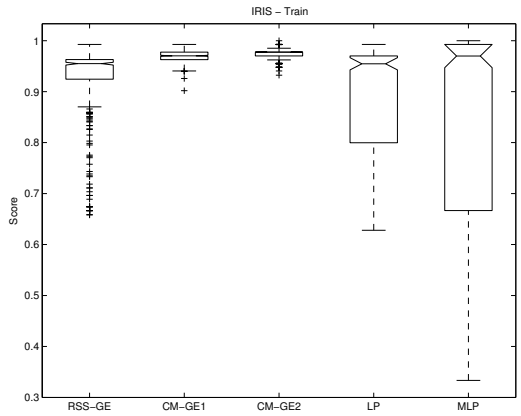


(e) Comparison of Mean Ranks (Train)

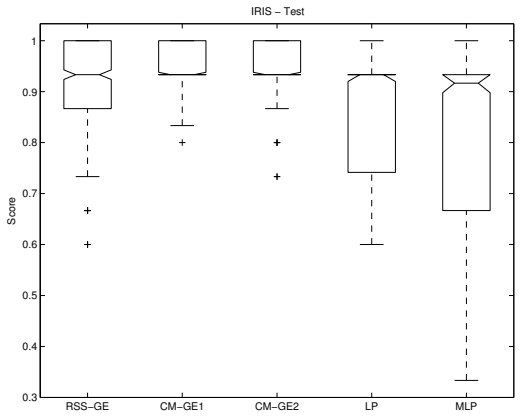


(f) Comparison of Mean Ranks (Test)

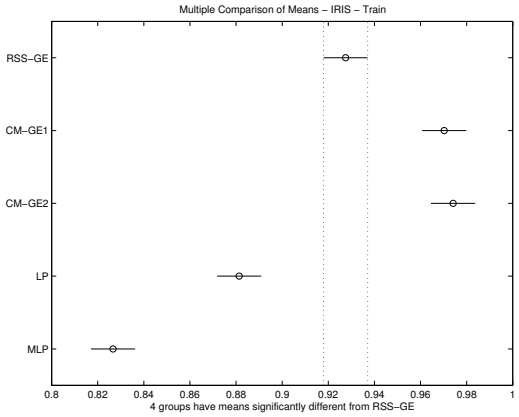
Figure B.17: ANN comparison of IMAG Score performance.



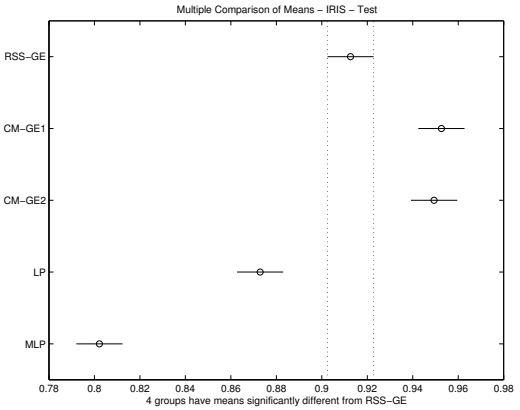
(a) Score (Train)



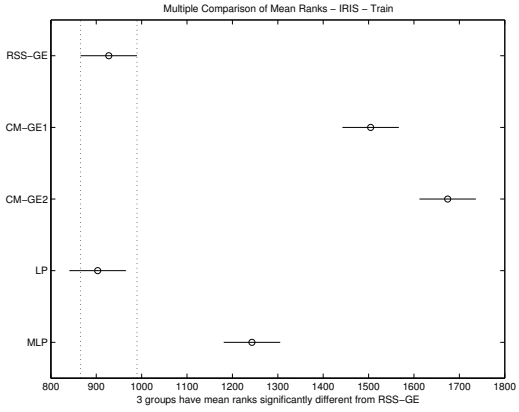
(b) Score (Test)



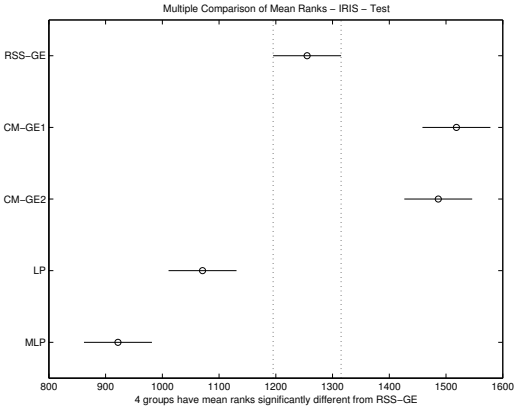
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

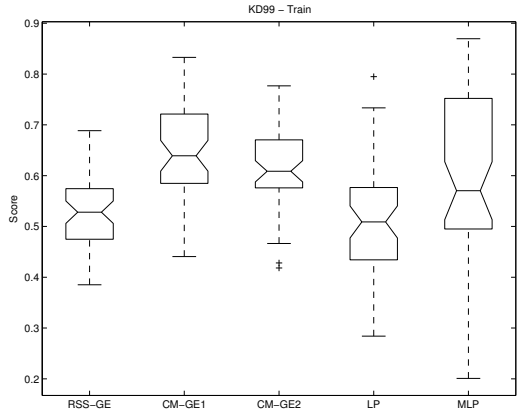


(e) Comparison of Mean Ranks (Train)

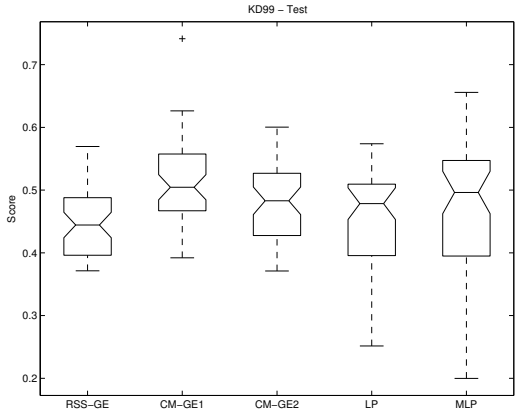


(f) Comparison of Mean Ranks (Test)

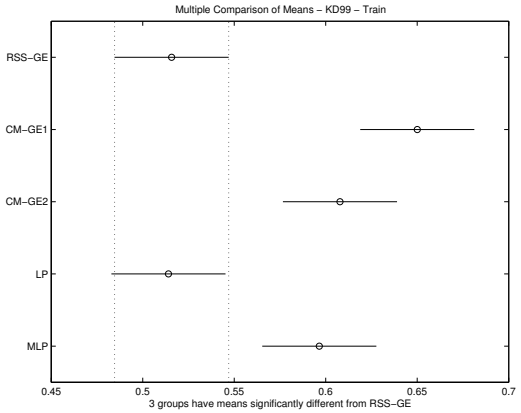
Figure B.18: ANN comparison of IRIS Score performance.



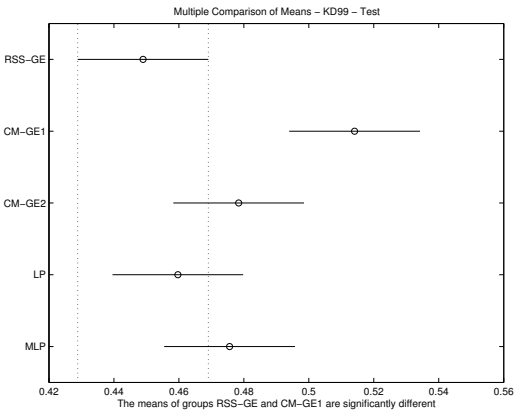
(a) Score (Train)



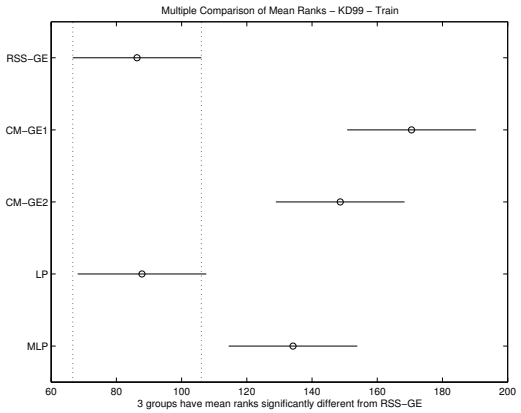
(b) Score (Test)



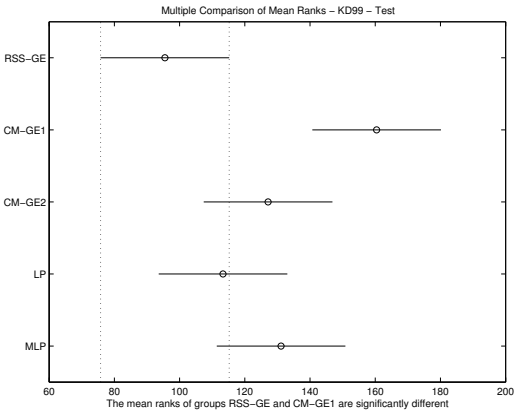
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

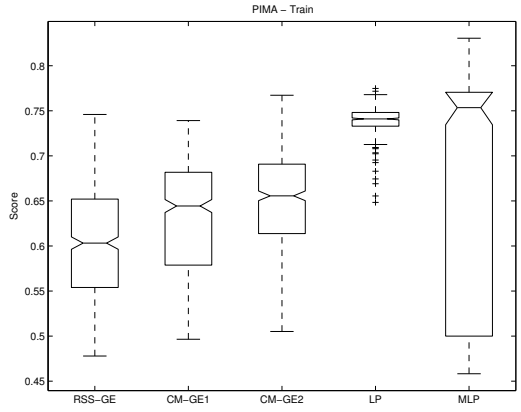


(e) Comparison of Mean Ranks (Train)

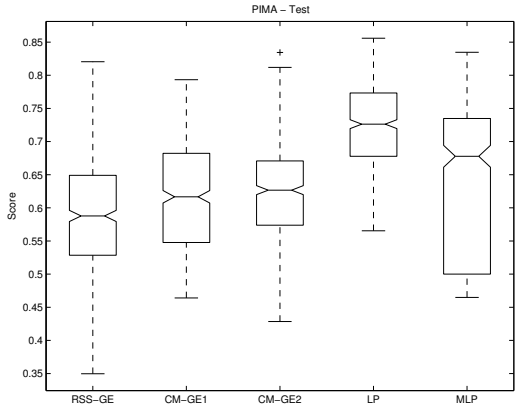


(f) Comparison of Mean Ranks (Test)

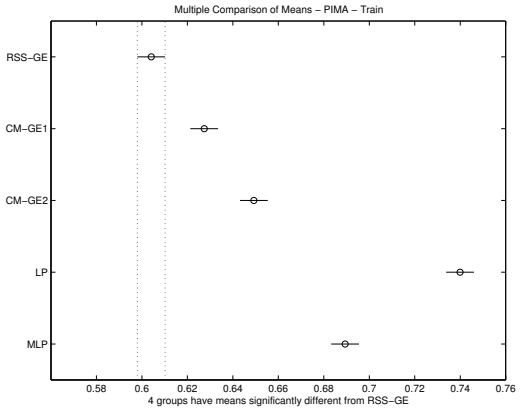
Figure B.19: ANN comparison of KD99 Score performance.



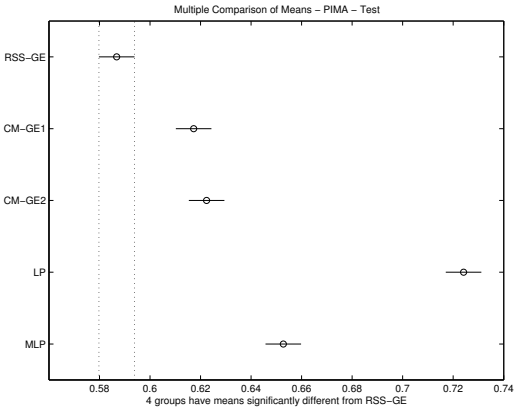
(a) Score (Train)



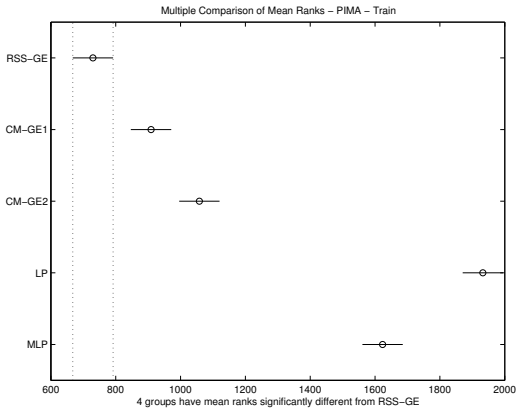
(b) Score (Test)



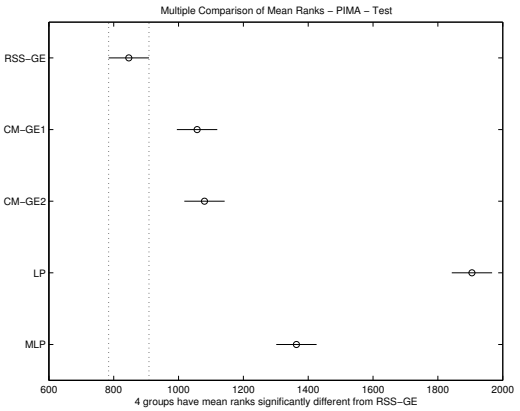
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

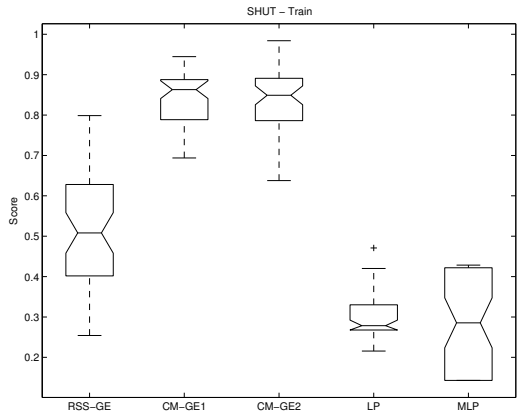


(e) Comparison of Mean Ranks (Train)

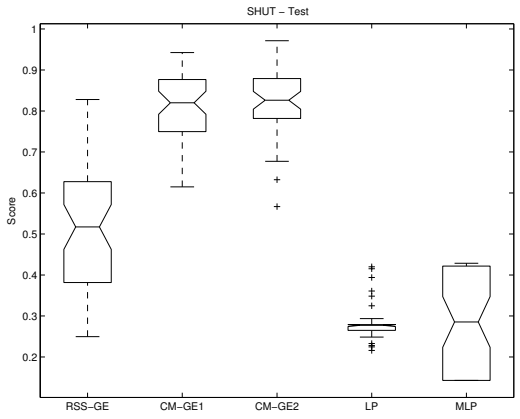


(f) Comparison of Mean Ranks (Test)

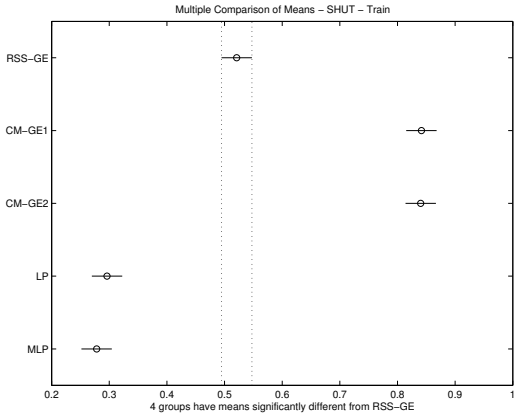
Figure B.20: ANN comparison of PIMA Score performance.



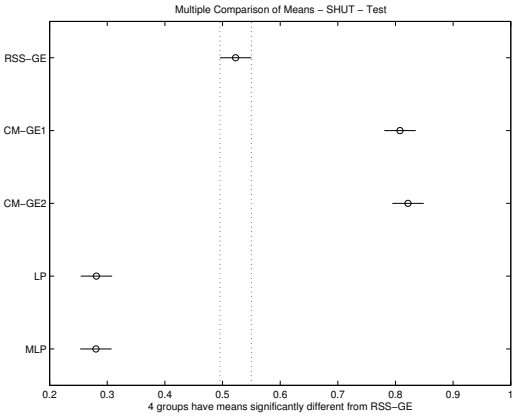
(a) Score (Train)



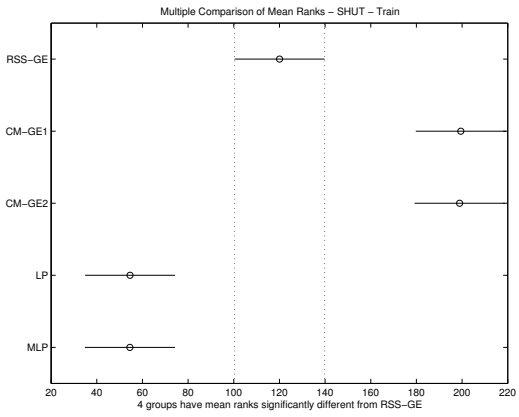
(b) Score (Test)



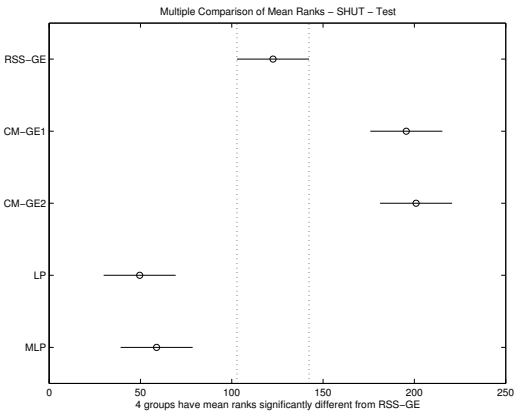
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

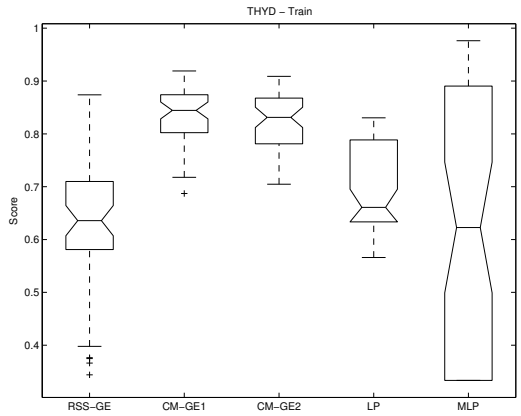


(e) Comparison of Mean Ranks (Train)

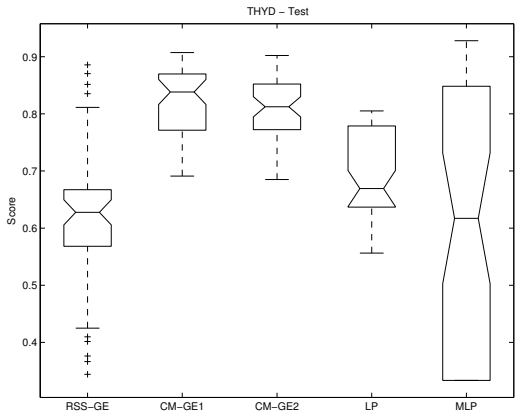


(f) Comparison of Mean Ranks (Test)

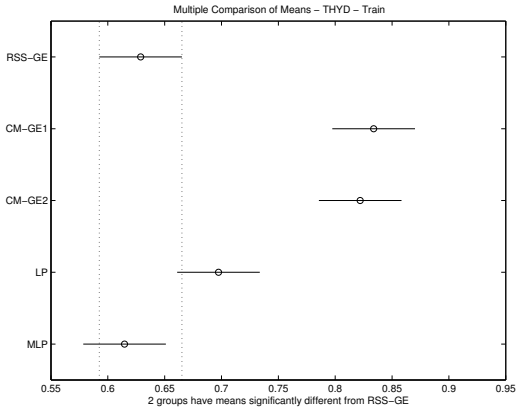
Figure B.21: ANN comparison of SHUT Score performance.



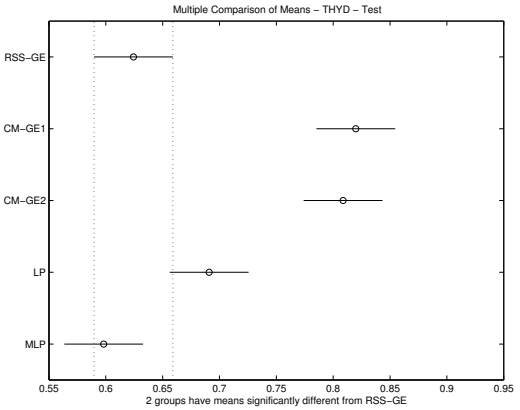
(a) Score (Train)



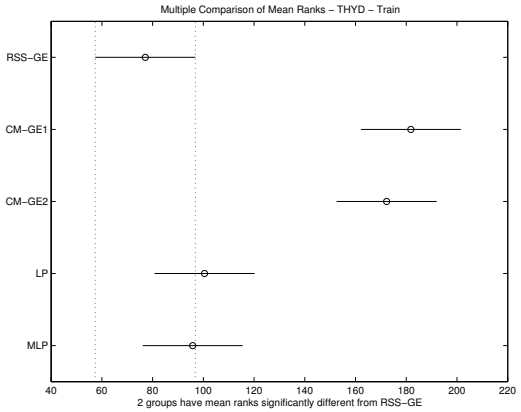
(b) Score (Test)



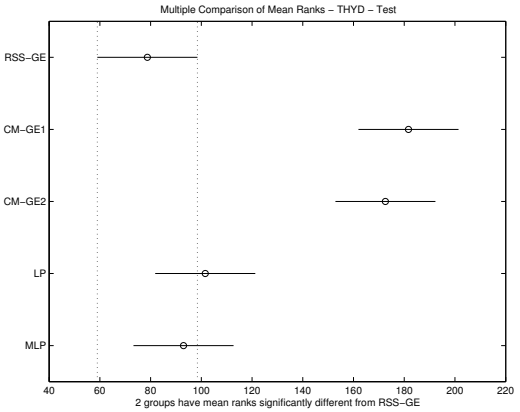
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

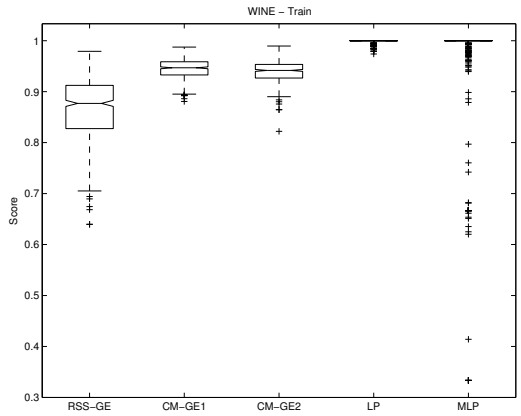


(e) Comparison of Mean Ranks (Train)

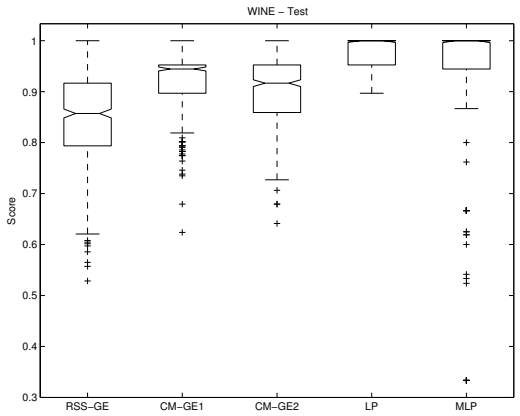


(f) Comparison of Mean Ranks (Test)

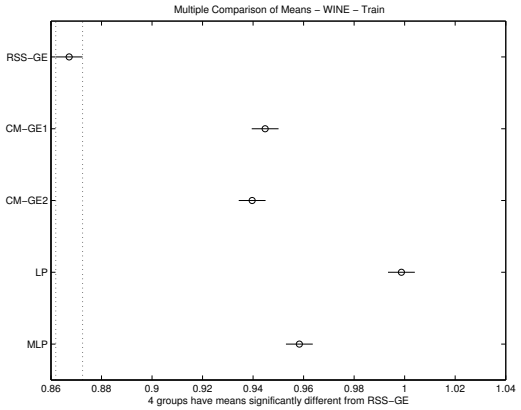
Figure B.22: ANN comparison of THYD Score performance.



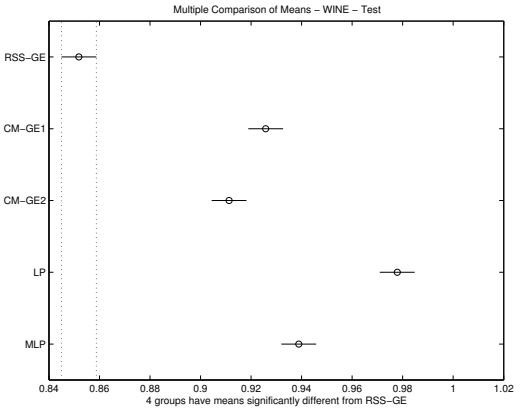
(a) Score (Train)



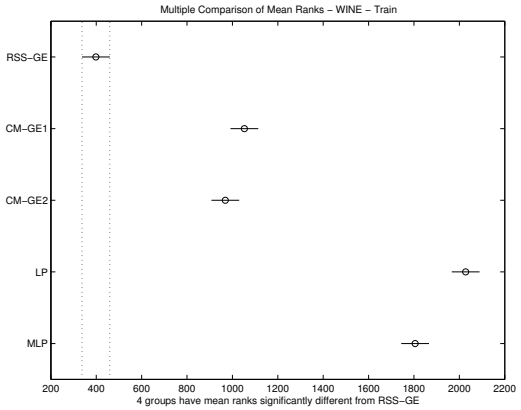
(b) Score (Test)



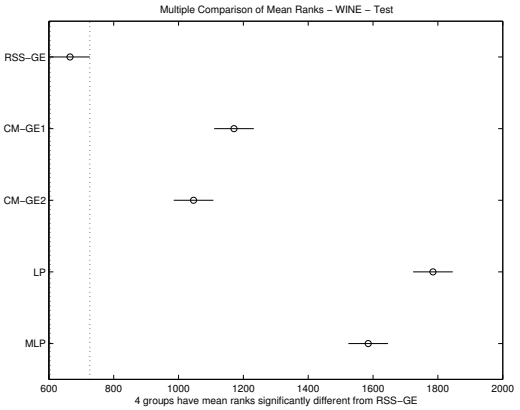
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)

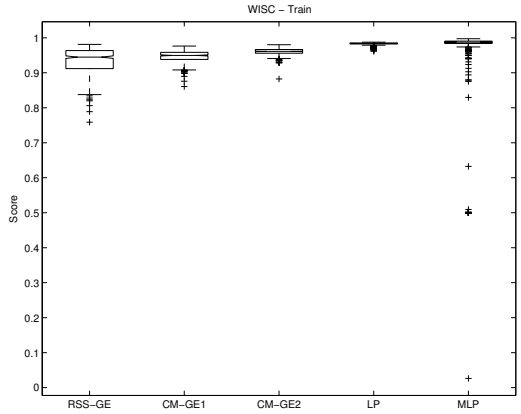


(e) Comparison of Mean Ranks (Train)

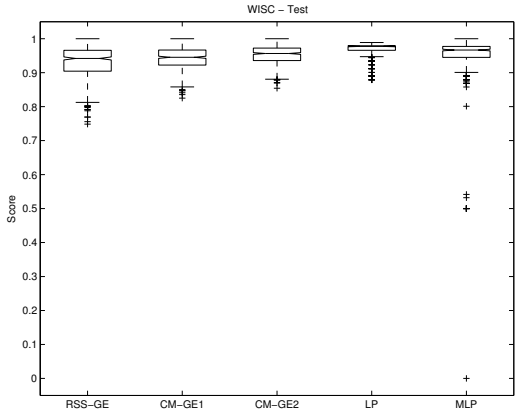


(f) Comparison of Mean Ranks (Test)

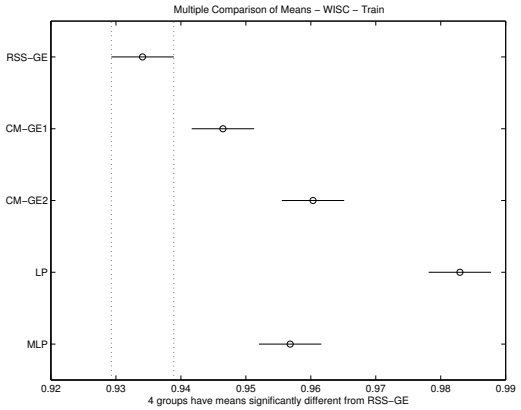
Figure B.23: ANN comparison of WINE Score performance.



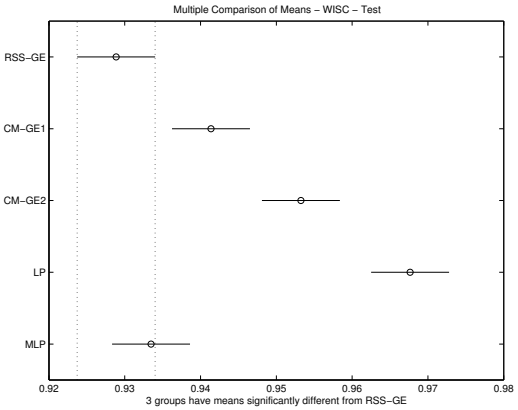
(a) Score (Train)



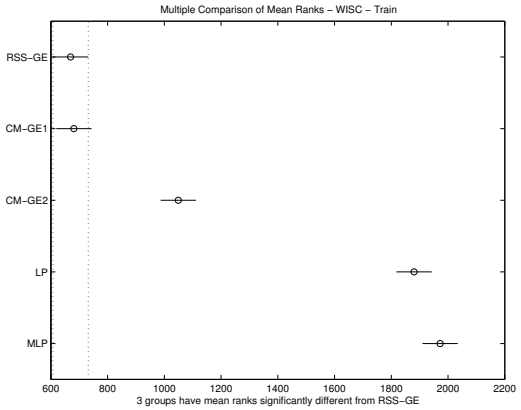
(b) Score (Test)



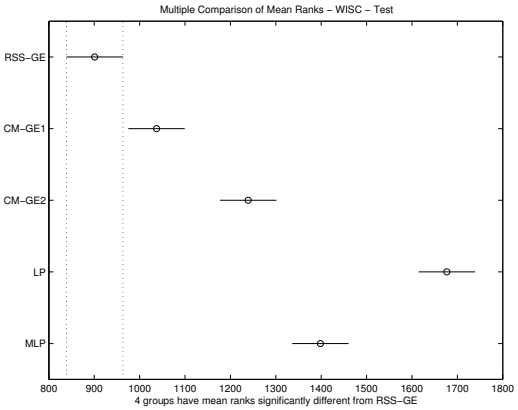
(c) Comparison of Means (Train)



(d) Comparison of Means (Test)



(e) Comparison of Mean Ranks (Train)



(f) Comparison of Mean Ranks (Test)

Figure B.24: ANN comparison of WISC Score performance.

Appendix C

Deterministic Comparisons (E8 - E10)

C.1 Training

Table C.1: Deterministic Comparison: J48 (Train)

Rss-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0	0	0.6	0	0	1.6	0	13.4	-	-	-	-	-	-	-	-
WISC	0	0	0.8	12	12	0.8	-	-	-	-	-	-	-	-	-	-
CENS	0	0	0	90	90	0	-	-	-	-	-	-	-	-	-	-
CONT	0	0	0.4	13.4	17.4	1.8	0.4	9.6	-	-	-	-	-	-	-	-
IMAG	0	0	6	22	58	16	0	12	0	2	4	14	10	6	78	100
IRIS	3.8	2.6	95.2	94.2	12.4	**43.6**	**43.2**	12.4	-	-	-	-	-	-	-	-
KD99	0	0	0	0	0	0	0	0	0	0	8	2	-	-	-	-
BUPA	0	0	13.2	0	0	13.2	-	-	-	-	-	-	-	-	-	-
PIMA	0	0	0.4	2.2	2.2	0.4	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	0	8	20	6	20	0	0	0	0	**28**	10	50	**30**	18
THYD	0	0	6	4	4	6	0	**30**	-	-	-	-	-	-	-	-
WINE	0	0	25	0.8	0.4	11.2	**47.8**	24.4	-	-	-	-	-	-	-	-
Std-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0	0	1.8	0.4	0.6	8.6	2.2	10	-	-	-	-	-	-	-	-
WISC	0	0.2	0.8	4.2	4.2	0.8	-	-	-	-	-	-	-	-	-	-
CENS	0	0	100	0	0	100	-	-	-	-	-	-	-	-	-	-
CONT	0	0	**47.4**	1.8	0.4	65.2	1.8	**47.8**	-	-	-	-	-	-	-	-
IMAG	0	0	6	**34**	62	18	0	0	0	**28**	0	**38**	18	**30**	100	100
IRIS	9.8	8.8	98.8	98.2	7.2	**45.4**	**45.4**	7.4	-	-	-	-	-	-	-	-
KD99	0	0	2	0	0	52	0	4	0	**48**	0	**36**	-	-	-	-
BUPA	0	0	2.6	**36**	**36**	2.6	-	-	-	-	-	-	-	-	-	-
PIMA	0	0	83.2	0	0	83.2	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	0	0	0	62	0	58	0	**36**	0	18	6	56	0	52
THYD	0	0	0	94	0	94	98	0	-	-	-	-	-	-	-	-
WINE	0.2	0	23.6	1.6	2.2	8.8	52.8	**45.6**	-	-	-	-	-	-	-	-
CMGE1	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0	0	0.8	0	0	9.2	0	0	-	-	-	-	-	-	-	-
WISC	0	0	0	1.6	1.6	0	-	-	-	-	-	-	-	-	-	-
CENS	0	0	18	0	0	18	-	-	-	-	-	-	-	-	-	-
CONT	0	0	**35.8**	0.2	0.2	23.4	0	51.4	-	-	-	-	-	-	-	-
IMAG	0	0	22	4	90	2	4	4	0	**28**	2	**44**	18	0	88	**46**
IRIS	14.4	12.6	99.2	98.6	3	63.4	63.8	3	-	-	-	-	-	-	-	-
KD99	0	0	0	0	0	4	0	0	0	6	16	0	-	-	-	-
BUPA	0	0	0.4	24.2	24.2	0.4	-	-	-	-	-	-	-	-	-	-
PIMA	0	0	**45.8**	0	0	**45.8**	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	0	0	14	4	0	0	0	2	0	16	84	4	64	2
THYD	0	0	0	0	2	0	0	22	-	-	-	-	-	-	-	-
WINE	0.2	0	24.4	4.4	4	6.4	73	**44.8**	-	-	-	-	-	-	-	-
CMGE2	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0	0	1.8	0	0	12.6	1.2	0.2	-	-	-	-	-	-	-	-
WISC	0	0	0	5.4	5.4	0	-	-	-	-	-	-	-	-	-	-
CENS	0	0	20	2	2	20	-	-	-	-	-	-	-	-	-	-
CONT	0	0	62.2	0.2	0	**40.2**	0.2	65.4	-	-	-	-	-	-	-	-
IMAG	0	0	**26**	2	90	0	0	2	0	**26**	0	**42**	16	0	98	86
IRIS	25	21.8	99.4	98.6	5.8	70	69.8	5.8	-	-	-	-	-	-	-	-
KD99	0	0	0	0	0	6	0	0	0	6	20	6	-	-	-	-
BUPA	0	0	2	4.6	4.6	2	-	-	-	-	-	-	-	-	-	-
PIMA	0	0	19.6	0.8	0.8	19.6	-	-	-	-	-	-	-	-	-	-
SHUT	0	8	0	0	14	6	0	2	0	0	12	14	80	0	58	6
THYD	0	0	0	0	2	0	0	16	-	-	-	-	-	-	-	-
WINE	0.2	0.2	**48.8**	1.8	2.4	20.6	69.8	**43**	-	-	-	-	-	-	-	-

Table C.2: Deterministic Comparison: NB (Train)

Rss-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	7	6.8	**49.8**	23.6	**44.6**	19.8	10	70.8	-	-	-	-	-	-	-	-
WISC	22.6	21.2	**29.6**	**34.4**	**34.4**	**29.6**	-	-	-	-	-	-	-	-	-	-
CENS	10	**28**	10	90	90	10	-	-	-	-	-	-	-	-	-	-
CONT	2.8	2	**40.6**	**33.8**	**45.2**	**36.4**	16.4	51.4	-	-	-	-	-	-	-	-
IMAG	0	0	6	64	58	16	70	**30**	0	22	12	76	24	6	78	100
IRIS	**41**	**41**	95.2	94.2	24.8	53.4	53.4	**42**	-	-	-	-	-	-	-	-
KD99	72	0	86	22	24	**38**	16	**28**	6	78	0	92	-	-	-	-
BUPA	86.4	69.8	7.8	97.8	97.8	7.8	-	-	-	-	-	-	-	-	-	-
PIMA	0	0.8	1.8	**36.8**	**36.8**	1.8	-	-	-	-	-	-	-	-	-	-
SHUT	4	0	16	80	**34**	78	66	62	52	0	0	**28**	10	50	**30**	86
THYD	0	22	66	4	58	10	0	100	-	-	-	-	-	-	-	-
WINE	0	0	25	0.8	0.4	11.2	**47.8**	24.4	-	-	-	-	-	-	-	-
Std-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	11	10.4	**36.8**	**35**	**45.2**	**32.8**	19.8	55.2	-	-	-	-	-	-	-	-
WISC	**27.8**	20.8	51	19.2	19.2	51	-	-	-	-	-	-	-	-	-	-
CENS	100	0	100	0	0	100	-	-	-	-	-	-	-	-	-	-
CONT	3.2	0	95	2.8	1	98.4	14.4	70.4	-	-	-	-	-	-	-	-
IMAG	0	0	6	70	62	18	100	12	4	**44**	4	78	**36**	**30**	100	100
IRIS	56.4	56.4	98.8	98.2	22.6	63	63	**46.8**	-	-	-	-	-	-	-	-
KD99	58	0	90	14	0	88	14	20	0	96	0	94	-	-	-	-
BUPA	99.8	55.4	2	99.6	99.6	2	-	-	-	-	-	-	-	-	-	-
PIMA	0.2	0	97	0.2	0.2	97	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	100	0	0	92	0	96	0	**36**	2	18	6	56	0	86
THYD	0	0	0	100	0	98	100	0	-	-	-	-	-	-	-	-
WINE	0.2	0	23.6	1.6	2.2	8.8	52.8	**45.6**	-	-	-	-	-	-	-	-
CMGE1	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	**37.8**	**39**	97.2	1.8	11.6	94.2	**39.2**	68.2	-	-	-	-	-	-	-	-
WISC	11.2	6.4	67	10.4	10.4	67	-	-	-	-	-	-	-	-	-	-
CENS	94	0	94	0	0	94	-	-	-	-	-	-	-	-	-	-
CONT	24.8	1.2	97.6	0.8	4.6	93.4	5.4	90	-	-	-	-	-	-	-	-
IMAG	24	24	22	24	90	2	100	10	0	100	6	100	64	0	88	**46**
IRIS	88.2	88.2	99.2	98.6	14.8	89.6	89.8	**44.4**	-	-	-	-	-	-	-	-
KD99	94	8	94	8	64	60	10	**28**	2	100	2	86	-	-	-	-
BUPA	100	67	0	100	100	0	-	-	-	-	-	-	-	-	-	-
PIMA	3	0.8	88.8	1.6	1.6	88.8	-	-	-	-	-	-	-	-	-	-
SHUT	**32**	58	12	94	66	90	62	**38**	92	2	10	16	60	4	64	**32**
THYD	20	94	**46**	6	98	2	2	100	-	-	-	-	-	-	-	-
WINE	0.2	0	24.4	4.4	4	6.4	73	**44.8**	-	-	-	-	-	-	-	-
CMGE2	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	**44.6**	**45.2**	91.4	7.4	19.4	86.8	**42**	62.6	-	-	-	-	-	-	-	-
WISC	**37.2**	**25.2**	56	**32.8**	**32.8**	56	-	-	-	-	-	-	-	-	-	-
CENS	94	0	94	2	2	94	-	-	-	-	-	-	-	-	-	-
CONT	21	0.4	99.2	0.4	0.8	97	0.8	95.4	-	-	-	-	-	-	-	-
IMAG	12	12	**26**	**30**	90	0	100	6	0	92	0	100	72	0	98	86
IRIS	94	94	99.4	98.6	22.4	89.2	89.4	57.6	-	-	-	-	-	-	-	-
KD99	94	0	94	6	64	84	**36**	8	0	100	2	86	-	-	-	-
BUPA	100	97.8	0.4	100	100	0.4	-	-	-	-	-	-	-	-	-	-
PIMA	2.8	4	60.8	12.2	12.2	60.8	-	-	-	-	-	-	-	-	-	-
SHUT	56	60	**26**	100	62	90	76	50	98	0	**32**	14	62	0	58	56
THYD	22	94	**44**	12	96	4	2	100	-	-	-	-	-	-	-	-
WINE	0.2	0.2	**48.8**	1.8	2.4	20.6	69.8	**43**	-	-	-	-	-	-	-	-

Table C.3: Deterministic Comparison: OneR (Train)

Rss-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0	0	**30.2**	4.4	18.2	18.6	10	**40.8**	-	-	-	-	-	-	-	-
WISC	61.6	74.2	2.8	90.4	90.4	2.8	-	-	-	-	-	-	-	-	-	-
CENS	0	100	0	100	100	0	-	-	-	-	-	-	-	-	-	-
CONT	8.2	86.8	1.2	96.8	99.4	0	13.6	80.4	-	-	-	-	-	-	-	-
IMAG	68	68	**44**	92	58	88	54	**38**	**40**	62	66	22	60	24	98	100
IRIS	**43.2**	**43.8**	95.2	94.2	56.2	53.4	53.4	57.6	-	-	-	-	-	-	-	-
KD99	4	76	22	100	100	0	**46**	24	**30**	22	100	2	-	-	-	-
BUPA	5	7.2	66.8	11.4	11.4	66.8	-	-	-	-	-	-	-	-	-	-
PIMA	0	5.6	0.4	76.6	76.6	0.4	-	-	-	-	-	-	-	-	-	-
SHUT	2	72	24	**44**	100	6	96	0	8	22	0	72	100	50	100	18
THYD	0	2	70	12	22	14	0	66	-	-	-	-	-	-	-	-
WINE	66.2	68	**37.6**	67.6	69.8	80	80.4	64.6	-	-	-	-	-	-	-	-
Std-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	1	1	21.6	20.6	24.4	**32.2**	19.8	**42**	-	-	-	-	-	-	-	-
WISC	71	79.2	7	85.2	85.2	7	-	-	-	-	-	-	-	-	-	-
CENS	0	0	100	0	0	100	-	-	-	-	-	-	-	-	-	-
CONT	7	24.2	67	**25.6**	55.2	**43.8**	13.6	80.4	-	-	-	-	-	-	-	-
IMAG	70	70	**28**	96	62	74	98	**32**	**38**	64	58	**48**	54	**44**	100	100
IRIS	57.6	58.4	98.8	98.2	64.4	63	63	64.8	-	-	-	-	-	-	-	-
KD99	0	2	20	54	100	52	**42**	16	2	82	100	**36**	-	-	-	-
BUPA	19.4	10.2	12.8	83	83	12.8	-	-	-	-	-	-	-	-	-	-
PIMA	0.6	1.2	86.8	4	4	86.8	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	100	0	100	62	6	58	0	90	2	58	100	56	100	52
THYD	0	0	0	100	0	100	100	0	-	-	-	-	-	-	-	-
WINE	76.8	77.2	**37.8**	76.4	79	73.4	82.8	82.2	-	-	-	-	-	-	-	-
CMGE1	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0	0	80.6	0	0.6	91.8	**39.2**	23.4	-	-	-	-	-	-	-	-
WISC	93	94.6	7.2	95.6	95.6	7.2	-	-	-	-	-	-	-	-	-	-
CENS	0	50	10	52	52	10	-	-	-	-	-	-	-	-	-	-
CONT	**44.6**	78.6	67.8	**32**	98.8	2.6	4.4	96	-	-	-	-	-	-	-	-
IMAG	96	96	78	98	90	64	100	**30**	**36**	100	66	70	98	18	96	92
IRIS	90.6	90.8	99.2	98.6	75.8	89.6	89.8	76.2	-	-	-	-	-	-	-	-
KD99	**40**	98	**34**	98	100	4	76	24	68	84	100	0	-	-	-	-
BUPA	**36.6**	23	5.8	96.2	96.2	5.8	-	-	-	-	-	-	-	-	-	-
PIMA	6.4	8.8	54.6	22.8	22.8	54.6	-	-	-	-	-	-	-	-	-	-
SHUT	18	100	**28**	**32**	100	4	98	0	4	100	6	82	100	4	100	2
THYD	2	**46**	62	66	54	22	14	54	-	-	-	-	-	-	-	-
WINE	100	100	**46.2**	99.6	100	96.6	99.8	97.2	-	-	-	-	-	-	-	-
CMGE2	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	0.2	0.2	67.2	1.2	3.6	85.4	**42**	**28.2**	-	-	-	-	-	-	-	-
WISC	98.8	99.8	4.2	99.8	99.8	4.2	-	-	-	-	-	-	-	-	-	-
CENS	0	50	12	52	52	12	-	-	-	-	-	-	-	-	-	-
CONT	**45.8**	70.6	88.4	12.2	98.6	2.4	0.8	99.2	-	-	-	-	-	-	-	-
IMAG	100	100	70	100	90	52	100	**28**	16	100	86	76	98	2	100	100
IRIS	95.2	95.2	99.4	98.6	87	89.2	89.4	87	-	-	-	-	-	-	-	-
KD99	**46**	94	**34**	100	100	6	84	2	**34**	88	100	6	-	-	-	-
BUPA	80.2	68.4	23.8	91	91	23.8	-	-	-	-	-	-	-	-	-	-
PIMA	5.6	19	**26.4**	**41.6**	**41.6**	**26.4**	-	-	-	-	-	-	-	-	-	-
SHUT	**44**	100	50	50	100	6	98	2	10	100	**30**	84	100	0	100	6
THYD	8	**36**	66	54	**38**	20	22	**44**	-	-	-	-	-	-	-	-
WINE	100	99.8	70.8	97.6	99.4	96.6	98.8	97.8	-	-	-	-	-	-	-	-

C.2 Test

Table C.4: Deterministic Comparison: J48 (Test)

Rss-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	7.2	6.6	**42.8**	21.8	24.2	**42**	14.4	**46.2**	-	-	-	-	-	-	-	-
WISC	22.2	**29.8**	**28**	50.6	50.6	**28**	-	-	-	-	-	-	-	-	-	-
CENS	0	74	0	92	92	0	-	-	-	-	-	-	-	-	-	-
CONT	9.8	23.2	9.2	63.2	82.6	16.4	**26.4**	51.8	-	-	-	-	-	-	-	-
IMAG	0	0	22	18	22	4	6	**28**	0	8	8	**38**	6	4	92	66
IRIS	60.6	67.4	98.2	98.6	59	67.6	67.4	59.2	-	-	-	-	-	-	-	-
KD99	0	0	**36**	16	10	0	0	18	2	58	**48**	0	-	-	-	-
BUPA	12.4	14.8	78.4	4	4	78.4	-	-	-	-	-	-	-	-	-	-
PIMA	4.6	9	5	**35.4**	**35.4**	9	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	0	8	**26**	8	20	0	0	0	0	56	100	60	**32**	24
THYD	0	0	**34**	4	4	8	0	52	-	-	-	-	-	-	-	-
WINE	21.8	24.2	56.2	**29**	17.8	54.6	78.8	56.8	-	-	-	-	-	-	-	-
Std-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	7.4	7.2	**36.2**	**34.4**	24.8	**45.8**	22.2	**43.4**	-	-	-	-	-	-	-	-
WISC	**26.4**	**31.8**	**37.8**	**42.6**	**42.6**	**37.8**	-	-	-	-	-	-	-	-	-	-
CENS	0	0	100	0	0	100	-	-	-	-	-	-	-	-	-	-
CONT	9.6	6	74.8	8.2	14.2	91.4	17.2	68.4	-	-	-	-	-	-	-	-
IMAG	0	0	12	22	20	8	16	14	0	**36**	0	60	10	20	100	72
IRIS	62.4	69.2	99.8	99	61.6	70	69.8	62.2	-	-	-	-	-	-	-	-
KD99	0	0	66	2	0	56	0	20	0	90	8	**48**	-	-	-	-
BUPA	18.6	15.4	**26.8**	67.6	67.6	**26.8**	-	-	-	-	-	-	-	-	-	-
PIMA	3	1.8	94.8	0.6	0.6	96.4	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	0	0	0	66	0	62	0	**42**	0	**30**	100	64	0	64
THYD	0	0	0	98	0	94	100	0	-	-	-	-	-	-	-	-
WINE	**25.4**	**29.6**	55.8	**31.6**	18.2	53	81	66.4	-	-	-	-	-	-	-	-
CMGE1	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	11	12.6	71.2	2.8	2.6	89.2	**34.8**	**33.4**	-	-	-	-	-	-	-	-
WISC	**32.8**	**37.2**	53.6	**35.6**	**35.6**	53.6	-	-	-	-	-	-	-	-	-	-
CENS	0	2	**44**	14	14	**44**	-	-	-	-	-	-	-	-	-	-
CONT	24.4	16	82	4.6	**38.2**	71.6	5.8	88.2	-	-	-	-	-	-	-	-
IMAG	0	0	**34**	0	**32**	0	14	14	0	52	2	80	16	0	94	**32**
IRIS	**47.2**	80.6	99.6	99.6	65.4	80	80.4	65.8	-	-	-	-	-	-	-	-
KD99	0	8	66	2	8	4	0	14	0	94	82	0	-	-	-	-
BUPA	23	15	21	74.2	74.2	21	-	-	-	-	-	-	-	-	-	-
PIMA	24.4	16.6	86.4	7.2	7.2	90.4	-	-	-	-	-	-	-	-	-	-
SHUT	0	**36**	0	0	16	12	4	0	0	2	4	**26**	100	4	70	2
THYD	0	0	6	4	20	0	0	**44**	-	-	-	-	-	-	-	-
WINE	23.6	58.2	61.6	54.2	**38.6**	59.6	93.2	77.6	-	-	-	-	-	-	-	-
CMGE2	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	10	11.4	65	7.8	4.6	84.2	**35**	**30.4**	-	-	-	-	-	-	-	-
WISC	**40.4**	**46.6**	**42.6**	54.2	54.2	**42.6**	-	-	-	-	-	-	-	-	-	-
CENS	0	4	58	6	6	58	-	-	-	-	-	-	-	-	-	-
CONT	20.8	9	92.4	1.8	**25.8**	82.6	2	95	-	-	-	-	-	-	-	-
IMAG	0	0	**30**	0	**26**	0	8	4	0	64	0	88	10	0	100	54
IRIS	**44.8**	78.8	99.2	99.6	67.8	75	75.6	68.2	-	-	-	-	-	-	-	-
KD99	0	0	58	0	2	6	0	0	0	96	68	6	-	-	-	-
BUPA	**31.8**	24.8	**42.4**	51.6	51.6	**42.4**	-	-	-	-	-	-	-	-	-	-
PIMA	16	11.4	61.8	10	10	67.8	-	-	-	-	-	-	-	-	-	-
SHUT	0	**40**	0	0	16	12	0	2	0	4	14	**30**	100	4	78	**30**
THYD	0	0	2	4	10	0	2	**30**	-	-	-	-	-	-	-	-
WINE	18	**46.4**	72	**37.8**	**26.2**	70.4	89.2	72.8	-	-	-	-	-	-	-	-

Table C.5: Deterministic Comparison: NB (Test)

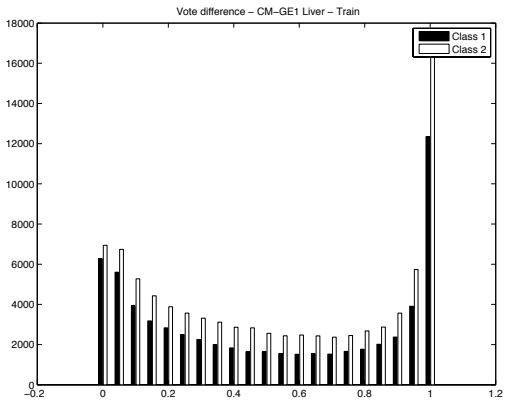
Rss-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	14.8	15	62	**33.2**	**47.2**	**38.4**	14.4	70.8	-	-	-	-	-	-	-	-
WISC	20.2	19	**28**	50.6	50.6	**28**	-	-	-	-	-	-	-	-	-	-
CENS	10	20	10	90	90	10	-	-	-	-	-	-	-	-	-	-
CONT	13	12.8	**47.6**	**40.8**	**45.4**	**47.4**	**26.4**	56	-	-	-	-	-	-	-	-
IMAG	0	0	**36**	**28**	**38**	2	76	20	0	**38**	4	80	**28**	4	60	**44**
IRIS	**31**	**31**	98.2	98.6	59	67.6	67.4	59.2	-	-	-	-	-	-	-	-
KD99	**28**	0	80	16	**46**	24	**36**	20	2	82	4	94	-	-	-	-
BUPA	65	60	**26.6**	82.6	82.6	**26.6**	-	-	-	-	-	-	-	-	-	-
PIMA	5.2	9.2	5	**35.4**	**35.4**	5	-	-	-	-	-	-	-	-	-	-
SHUT	4	6	14	80	60	74	64	64	52	0	0	56	10	60	**32**	86
THYD	0	22	58	10	62	8	0	100	-	-	-	-	-	-	-	-
WINE	6.6	6.6	56.2	**29**	17.8	54.6	78.8	56.8	-	-	-	-	-	-	-	-
Std-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	16	15.2	52.8	**46**	**44.4**	**43**	22.2	57.2	-	-	-	-	-	-	-	-
WISC	25	21	**37.8**	**42.6**	**42.6**	**37.8**	-	-	-	-	-	-	-	-	-	-
CENS	100	0	100	0	0	100	-	-	-	-	-	-	-	-	-	-
CONT	12.8	1.6	94.8	3.8	2	98.2	17.2	71	-	-	-	-	-	-	-	-
IMAG	0	0	22	**42**	**28**	4	100	0	0	52	0	80	**40**	24	78	56
IRIS	**36.8**	**36.8**	99.8	99	61.6	70	69.8	62.2	-	-	-	-	-	-	-	-
KD99	20	0	90	2	4	86	**28**	24	0	96	0	94	-	-	-	-
BUPA	88.8	52.6	5.4	98.6	98.6	5.4	-	-	-	-	-	-	-	-	-	-
PIMA	3.2	1.8	94.8	0.6	0.6	94.8	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	98	0	2	92	0	96	0	**42**	2	**30**	6	64	0	88
THYD	0	0	0	100	0	98	100	0	-	-	-	-	-	-	-	-
WINE	8.8	8.8	55.8	**31.6**	18.2	53	81	66.4	-	-	-	-	-	-	-	-
CMGE1	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	23.8	24	93	7.8	16.6	85.6	**34.8**	65.4	-	-	-	-	-	-	-	-
WISC	22.6	21.6	53.6	**35.6**	**35.6**	53.6	-	-	-	-	-	-	-	-	-	-
CENS	94	0	94	0	0	94	-	-	-	-	-	-	-	-	-	-
CONT	**29.6**	7	98.2	2.2	6.8	95.6	5.8	89.2	-	-	-	-	-	-	-	-
IMAG	4	4	52	2	84	0	100	0	0	100	0	100	72	0	62	22
IRIS	**47.2**	**47.2**	99.6	99.6	65.4	80	80.4	65.8	-	-	-	-	-	-	-	-
KD99	66	2	90	2	80	58	14	**38**	0	98	4	88	-	-	-	-
BUPA	92.8	60.8	1.6	100	100	1.6	-	-	-	-	-	-	-	-	-	-
PIMA	24.6	16.6	86.4	7.2	7.2	86.4	-	-	-	-	-	-	-	-	-	-
SHUT	**32**	60	10	94	84	90	64	**38**	92	2	14	**26**	70	4	70	**30**
THYD	16	90	**40**	20	96	0	0	100	-	-	-	-	-	-	-	-
WINE	20.2	20.2	61.6	54.2	**38.6**	59.6	93.2	77.6	-	-	-	-	-	-	-	-
CMGE2	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	**25.4**	**26.4**	88.6	14.8	20	80.2	**35**	59.6	-	-	-	-	-	-	-	-
WISC	**28.4**	**29.6**	**42.6**	54.2	54.2	**42.6**	-	-	-	-	-	-	-	-	-	-
CENS	94	0	94	0	0	94	-	-	-	-	-	-	-	-	-	-
CONT	**32**	2	98.6	0.6	1.8	97.4	2	95.6	-	-	-	-	-	-	-	-
IMAG	0	0	**40**	2	74	0	100	0	0	96	0	100	78	0	16	**26**
IRIS	**44.8**	**44.8**	99.2	99.6	67.8	75	75.6	68.2	-	-	-	-	-	-	-	-
KD99	50	0	90	0	70	76	20	12	0	100	8	82	-	-	-	-
BUPA	92.8	78.4	5.2	99.6	99.6	5.2	-	-	-	-	-	-	-	-	-	-
PIMA	16.6	12.6	61.8	10	10	61.8	-	-	-	-	-	-	-	-	-	-
SHUT	52	70	24	100	70	90	68	52	98	4	**40**	**30**	80	4	78	56
THYD	10	92	**34**	**26**	98	2	2	100	-	-	-	-	-	-	-	-
WINE	15.2	15.2	72	**37.8**	**26.2**	70.4	89.2	72.8	-	-	-	-	-	-	-	-

Table C.6: Deterministic Comparison: OneR (Test)

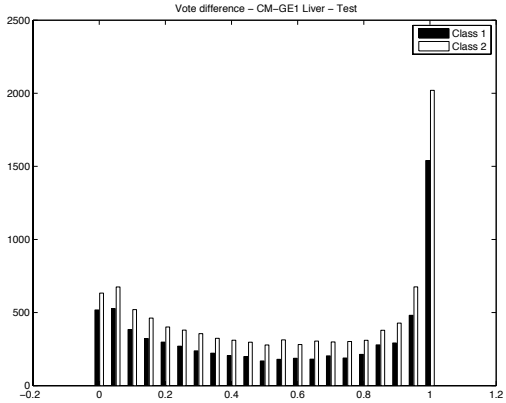
Rss-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	8.6	7.6	**36.8**	**26**	**33**	**33.4**	14.4	**46.2**	-	-	-	-	-	-	-	-
WISC	67	80.2	24	93.4	93.4	24	-	-	-	-	-	-	-	-	-	-
CENS	0	100	0	100	100	0	-	-	-	-	-	-	-	-	-	-
CONT	18.8	75.6	5	96	100	0.2	16.2	76.8	-	-	-	-	-	-	-	-
IMAG	88	88	**48**	90	22	**48**	58	**42**	**30**	66	74	**38**	68	**26**	98	100
IRIS	60.6	67.4	98.2	98.6	59	67.6	67.4	59.2	-	-	-	-	-	-	-	-
KD99	0	**28**	20	100	100	0	52	10	8	**28**	100	0	-	-	-	-
BUPA	76.8	77.6	89.4	**38.2**	**38.2**	89.4	-	-	-	-	-	-	-	-	-	-
PIMA	10	23.6	3	84.2	84.2	3	-	-	-	-	-	-	-	-	-	-
SHUT	2	70	24	**44**	100	8	92	12	10	22	6	72	100	60	100	24
THYD	0	8	70	14	**38**	14	0	72	-	-	-	-	-	-	-	-
WINE	79.4	70	56.2	76.4	67.2	75.6	90.4	56.8	-	-	-	-	-	-	-	-
Std-GE	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	8.8	8	**32.2**	**39.2**	**33.4**	**38**	22.2	**43.4**	-	-	-	-	-	-	-	-
WISC	70	80.4	**32.8**	86.8	86.8	**32.8**	-	-	-	-	-	-	-	-	-	-
CENS	0	0	100	0	0	100	-	-	-	-	-	-	-	-	-	-
CONT	18.4	**30.2**	67.4	24	60.2	**47.4**	13.8	80	-	-	-	-	-	-	-	-
IMAG	84	82	**32**	96	20	66	100	**36**	**38**	68	70	60	54	**46**	100	100
IRIS	62.4	69.2	99.8	99	61.6	70	69.8	62.2	-	-	-	-	-	-	-	-
KD99	0	0	**46**	50	100	56	**46**	8	2	80	100	**48**	-	-	-	-
BUPA	95.2	72	**38**	89.8	89.8	**38**	-	-	-	-	-	-	-	-	-	-
PIMA	7.6	5.4	91.2	7.8	7.8	91.2	-	-	-	-	-	-	-	-	-	-
SHUT	0	0	100	0	100	66	4	92	0	90	4	60	100	64	100	64
THYD	0	0	0	100	0	100	100	0	-	-	-	-	-	-	-	-
WINE	84.4	76.8	55.8	81.4	78.6	76.2	90.4	66.4	-	-	-	-	-	-	-	-
CMGE1	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	11	13.8	63.8	4.4	5.2	80.4	**34.8**	**33.4**	-	-	-	-	-	-	-	-
WISC	82.8	89.8	**45**	93.4	93.4	**45**	-	-	-	-	-	-	-	-	-	-
CENS	0	52	8	54	54	8	-	-	-	-	-	-	-	-	-	-
CONT	**39.4**	63.8	70.2	**29**	99.6	3.4	3	96.6	-	-	-	-	-	-	-	-
IMAG	98	98	72	98	**32**	**26**	100	**26**	6	100	74	80	100	10	96	96
IRIS	**47.2**	80.6	99.6	99.6	65.4	80	80.4	65.8	-	-	-	-	-	-	-	-
KD99	10	64	**36**	92	100	4	50	4	4	86	100	0	-	-	-	-
BUPA	96.8	79.2	**35.2**	97.6	97.6	**35.2**	-	-	-	-	-	-	-	-	-	-
PIMA	**45**	**35.2**	72.2	**40.8**	**40.8**	72.2	-	-	-	-	-	-	-	-	-	-
SHUT	18	100	**28**	**34**	100	12	98	10	6	100	82	84	100	4	100	2
THYD	4	58	60	68	62	16	12	62	-	-	-	-	-	-	-	-
WINE	98.8	94.8	61.6	96.2	94.2	86	98	77.6	-	-	-	-	-	-	-	-
CMGE2	**ACC**	**SCORE**	**DR1**	**FPR1**	**DR2**	**FPR2**	**DR3**	**FPR3**	**DR4**	**FPR4**	**DR5**	**FPR5**	**DR6**	**FPR6**	**DR7**	**FPR7**
BOST	10	12.4	58.2	10.6	10.8	75.2	**35**	**30.4**	-	-	-	-	-	-	-	-
WISC	90.6	97	**36.8**	99.2	99.2	**36.8**	-	-	-	-	-	-	-	-	-	-
CENS	0	52	12	56	56	12	-	-	-	-	-	-	-	-	-	-
CONT	**44.2**	59.4	87.8	12	99	4.8	1.2	98.4	-	-	-	-	-	-	-	-
IMAG	100	100	64	100	**26**	16	100	18	0	100	86	90	98	4	100	100
IRIS	**44.8**	78.8	99.2	99.6	67.8	75	75.6	68.2	-	-	-	-	-	-	-	-
KD99	8	**48**	**28**	94	100	6	76	0	0	90	100	6	-	-	-	-
BUPA	96.6	91.8	64.4	90.4	90.4	64.4	-	-	-	-	-	-	-	-	-	-
PIMA	**38**	**34.8**	**43.8**	54.6	54.6	**43.8**	-	-	-	-	-	-	-	-	-	-
SHUT	**44**	100	50	**46**	100	12	98	12	14	100	68	86	100	4	100	**30**
THYD	8	**42**	50	58	58	16	12	56	-	-	-	-	-	-	-	-
WINE	97	92.4	72	89.4	86	91	96.4	72.8	-	-	-	-	-	-	-	-

Appendix D

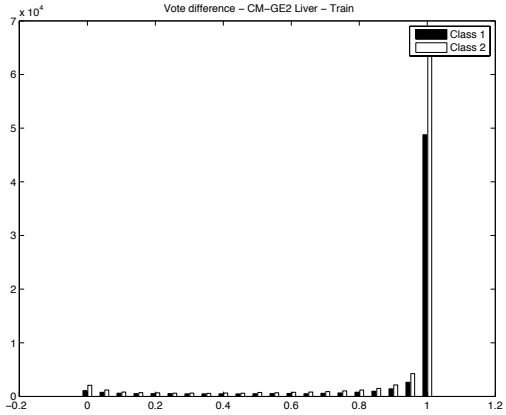
Coverage Results (E11 - E12)



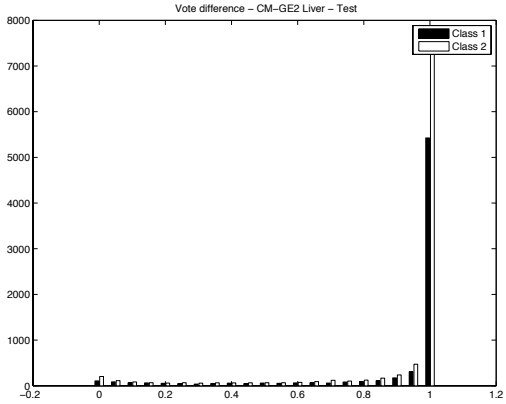
(a) CMGE1 Coverage (Train)



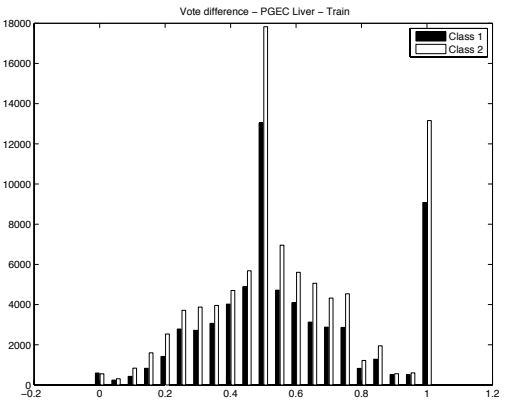
(b) CMGE1 Coverage (Test)



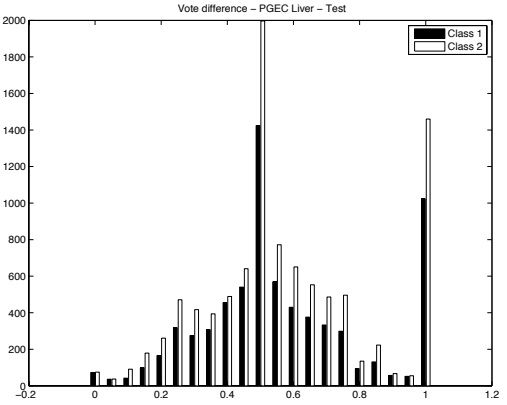
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)

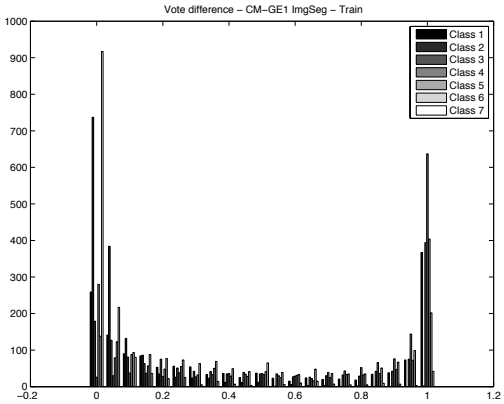


(e) PGEC Coverage (Train)

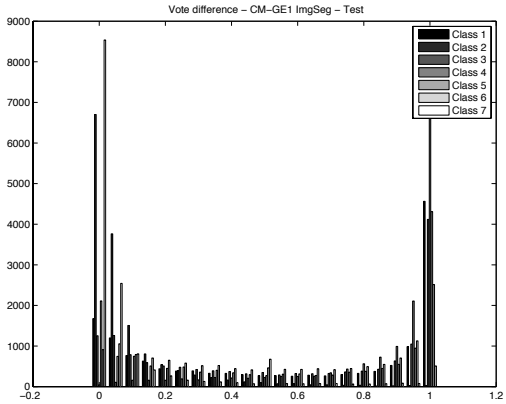


(f) PGEC Coverage (Test)

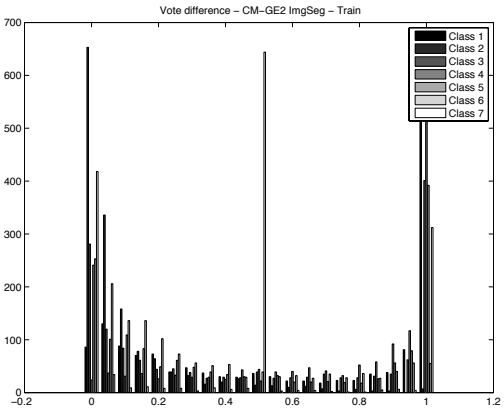
Figure D.1: Coverage comparison of CMGE 1, 2 and PGEC on BUPA.



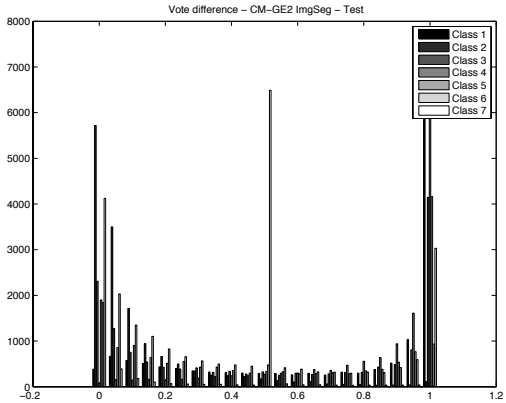
(a) CMGE1 Coverage (Train)



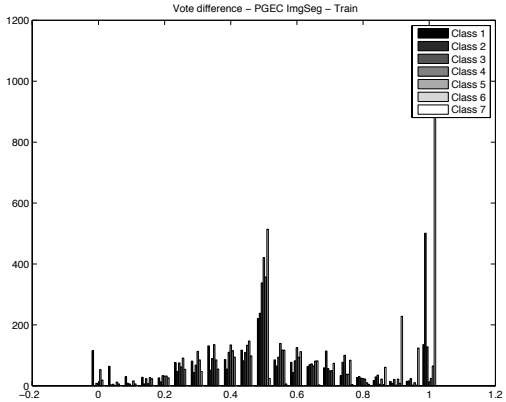
(b) CMGE1 Coverage (Test)



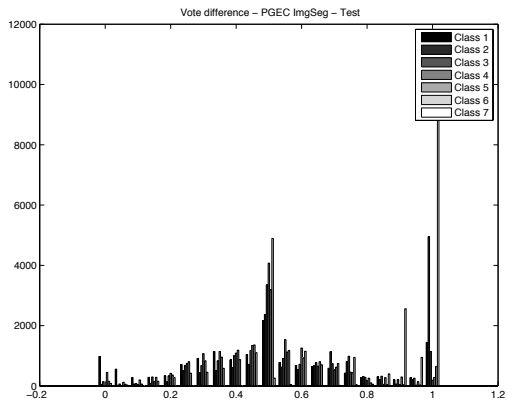
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)

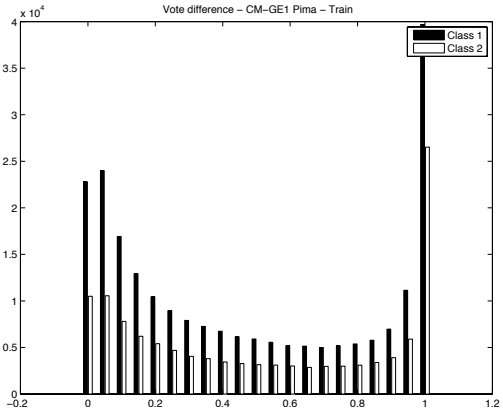


(e) PGEC Coverage (Train)

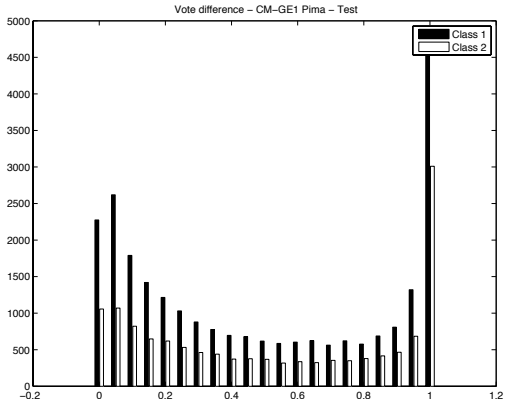


(f) PGEC Coverage (Test)

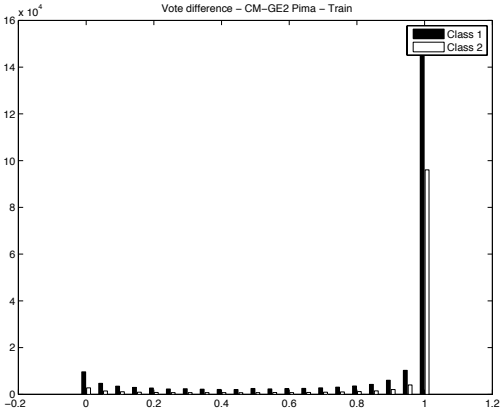
Figure D.2: Coverage comparison of CMGE 1, 2 and PGEC on IMAG.



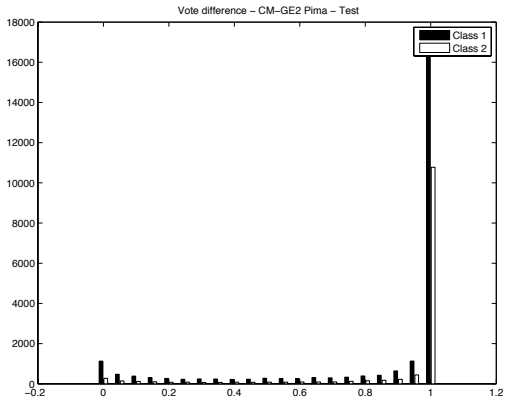
(a) CMGE1 Coverage (Train)



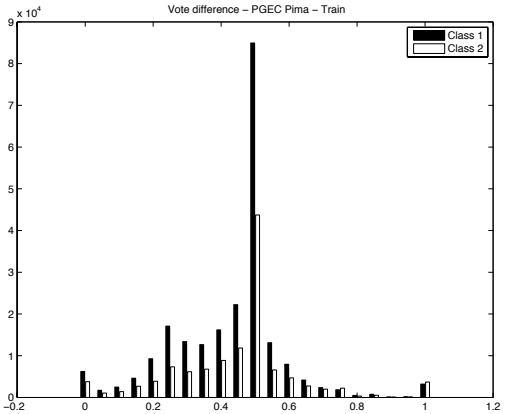
(b) CMGE1 Coverage (Test)



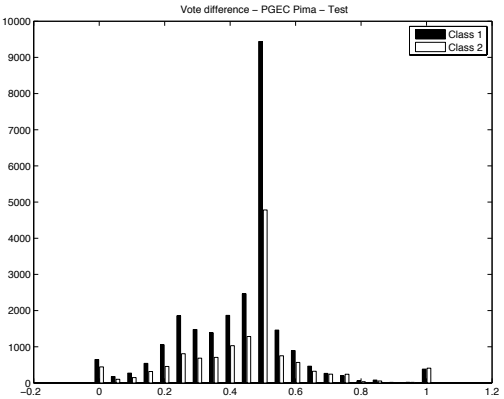
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)

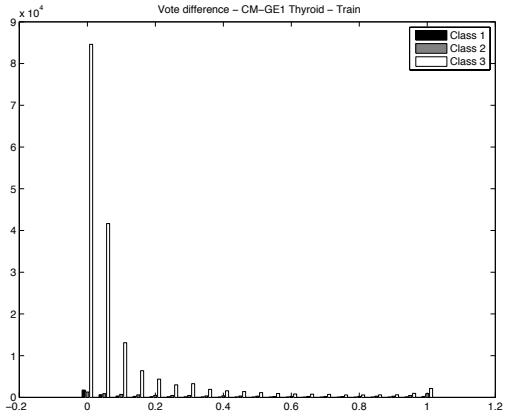


(e) PGEC Coverage (Train)

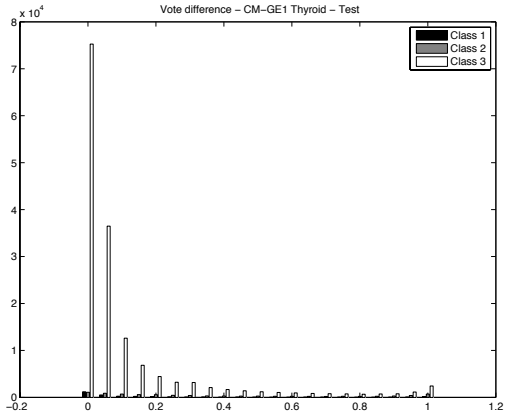


(f) PGEC Coverage (Test)

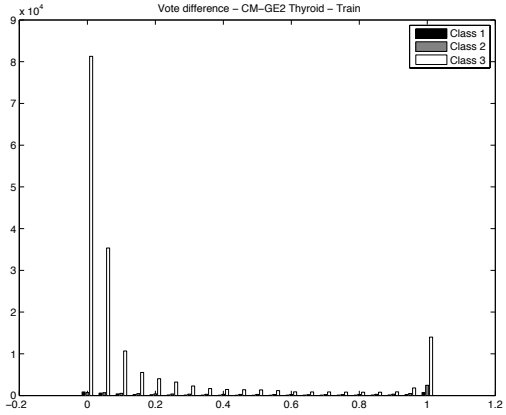
Figure D.3: Coverage comparison of CMGE 1, 2 and PGEC on PIMA.



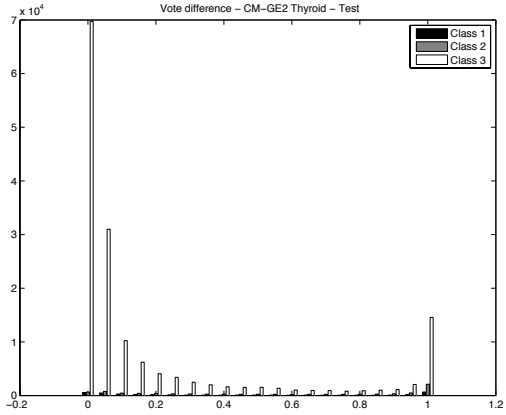
(a) CMGE1 Coverage (Train)



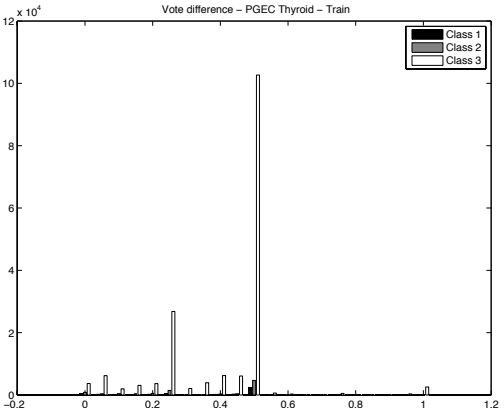
(b) CMGE1 Coverage (Test)



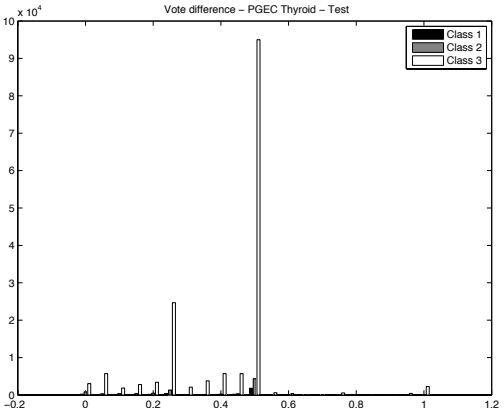
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)

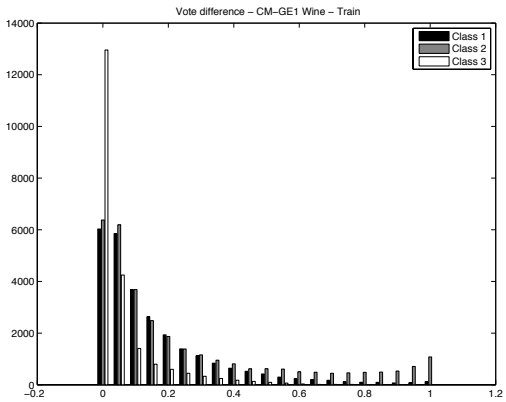


(e) PGEC Coverage (Train)

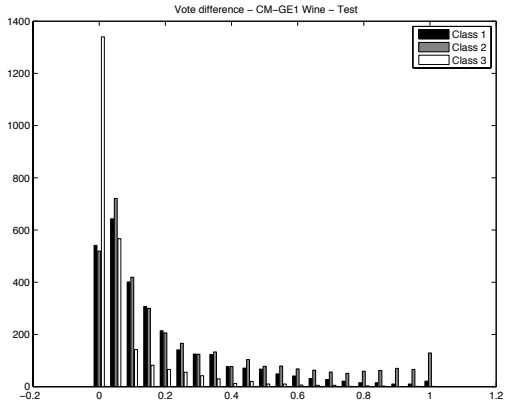


(f) PGEC Coverage (Test)

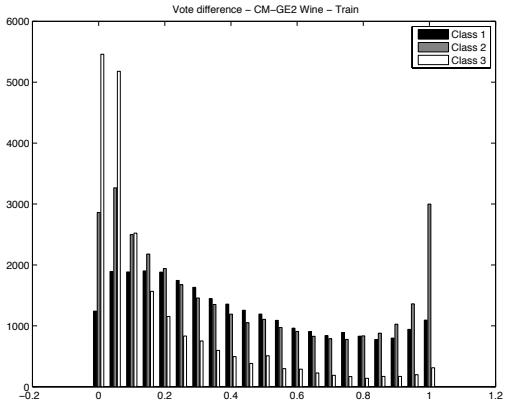
Figure D.4: Coverage comparison of CMGE 1, 2 and PGEC on THYD.



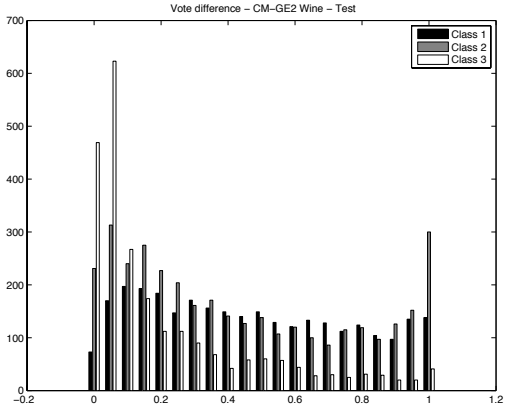
(a) CMGE1 Coverage (Train)



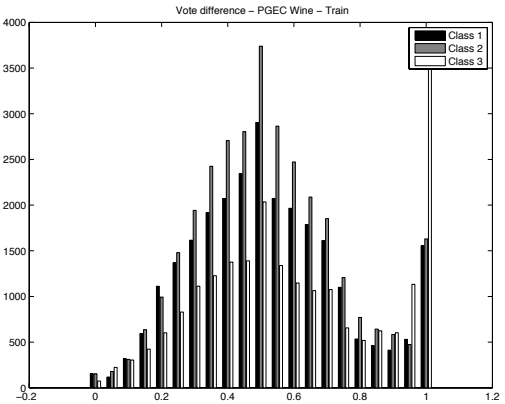
(b) CMGE1 Coverage (Test)



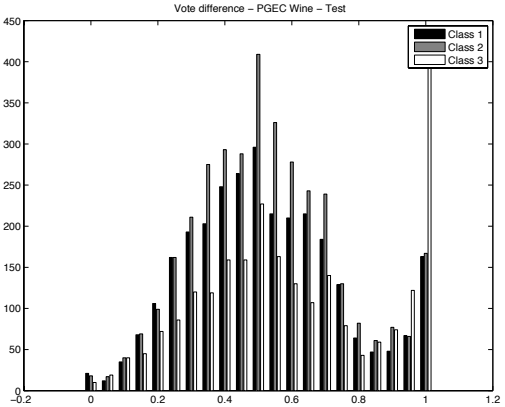
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)

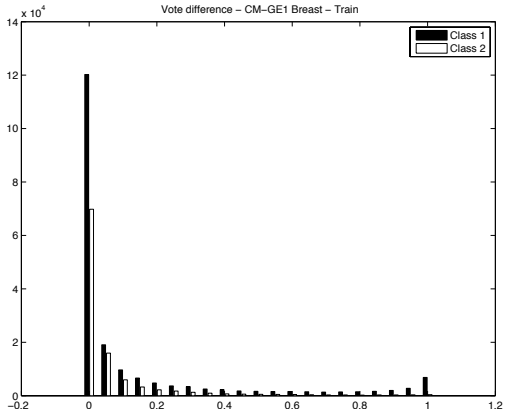


(e) PGEC Coverage (Train)

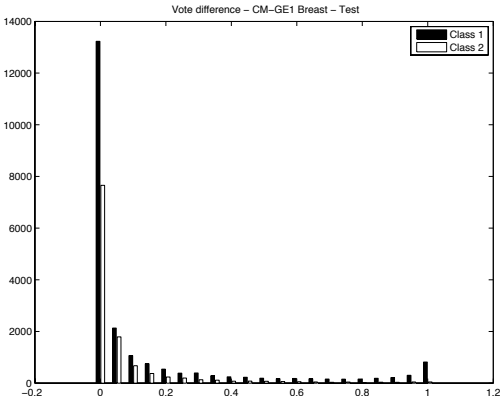


(f) PGEC Coverage (Test)

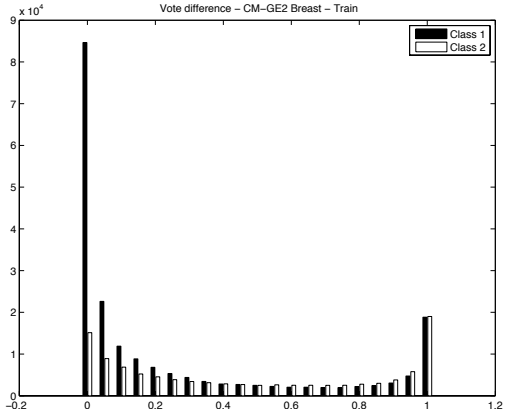
Figure D.5: Coverage comparison of CMGE 1, 2 and PGEC on WINE.



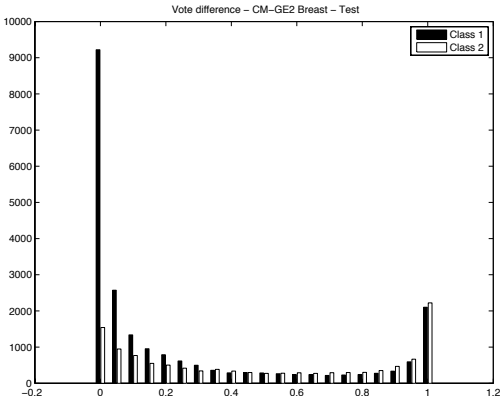
(a) CMGE1 Coverage (Train)



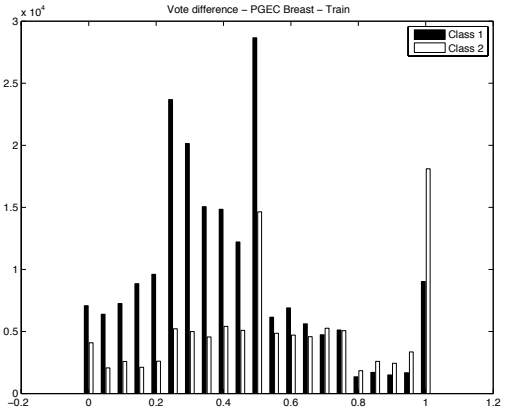
(b) CMGE1 Coverage (Test)



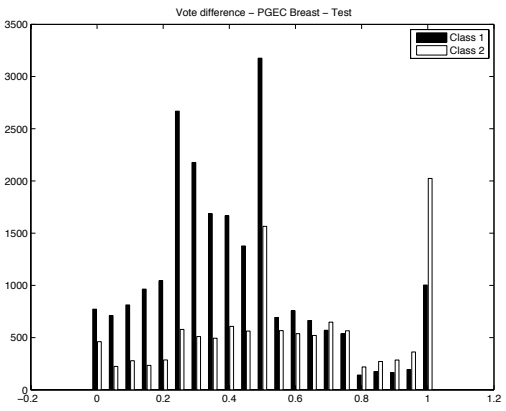
(c) CMGE2 Coverage (Train)



(d) CMGE2 Coverage (Test)



(e) PGEC Coverage (Train)

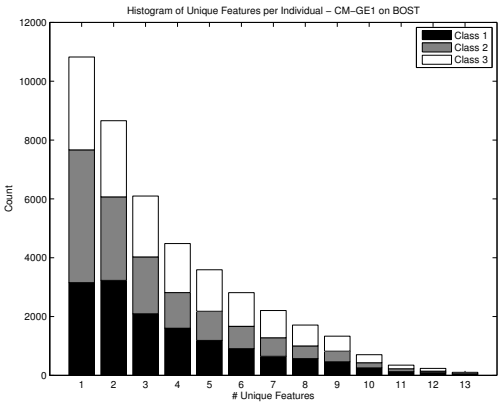


(f) PGEC Coverage (Test)

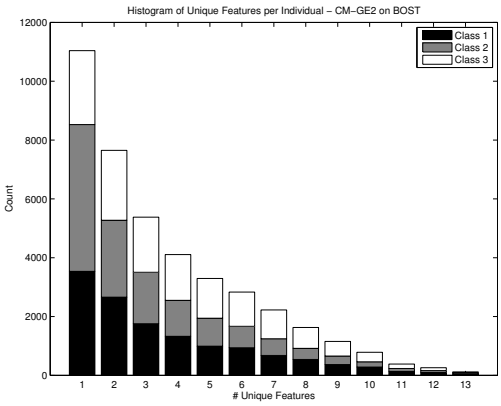
Figure D.6: Coverage comparison of CMGE 1, 2 and PGEC on WISC.

Appendix E

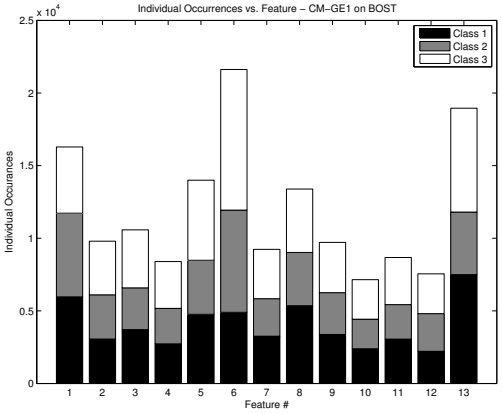
Results of Feature Analysis



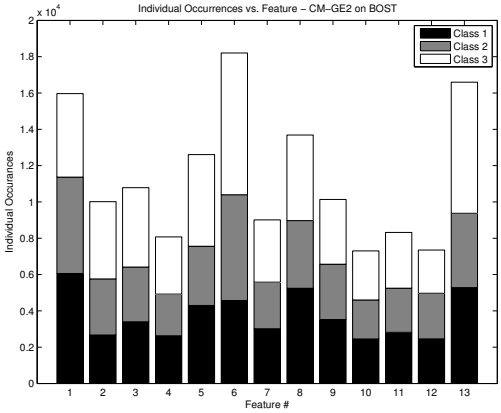
(a) CMGE1 Features per Individual



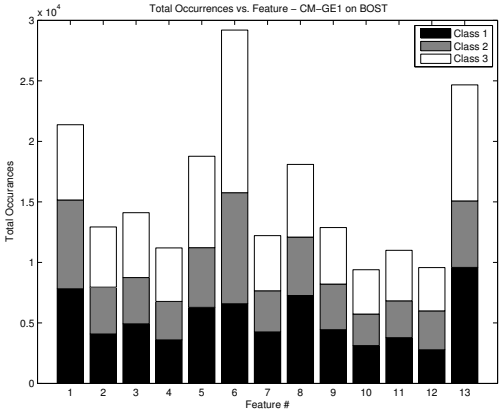
(b) CMGE2 Features per Individual



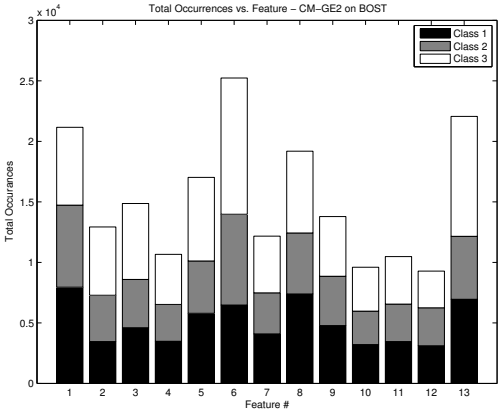
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

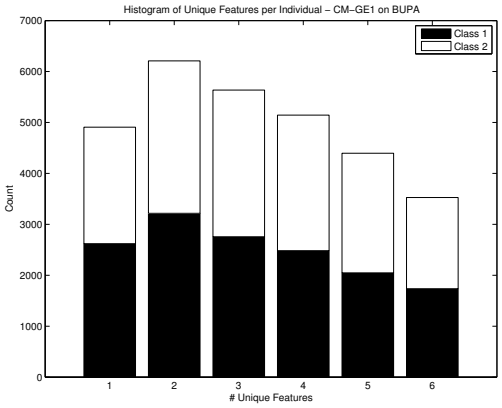


(e) CMGE1 Total Occurrences

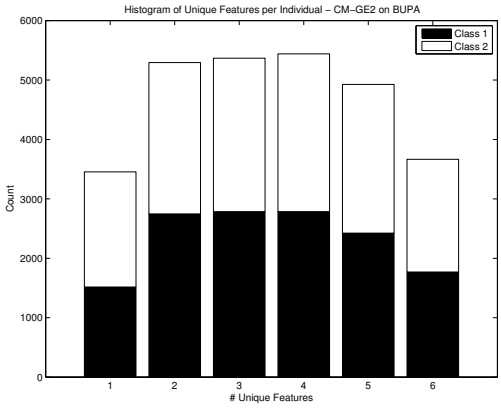


(f) CMGE2 Total Occurrences

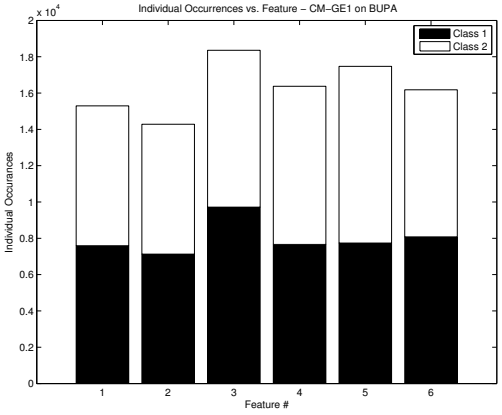
Figure E.1: Feature Analysis: CMGE 1, 2 on BOST.



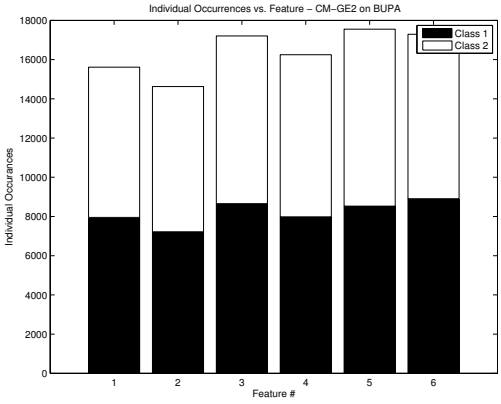
(a) CMGE1 Features per Individual



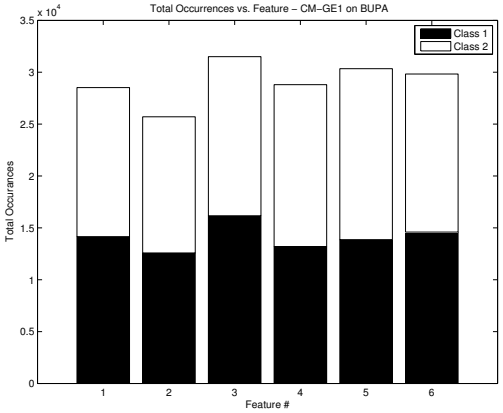
(b) CMGE2 Features per Individual



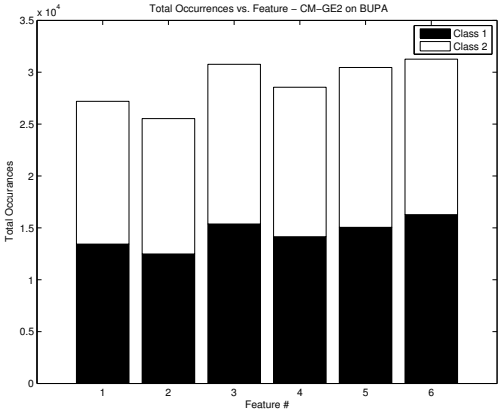
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

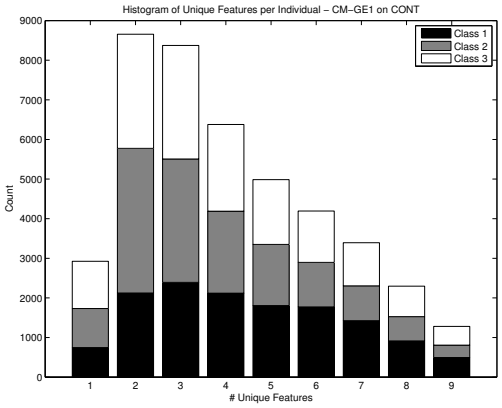


(e) CMGE1 Total Occurrences

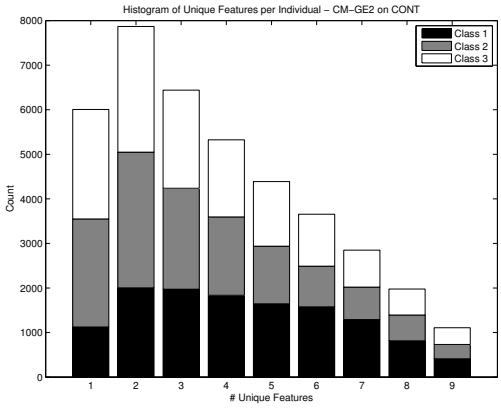


(f) CMGE2 Total Occurrences

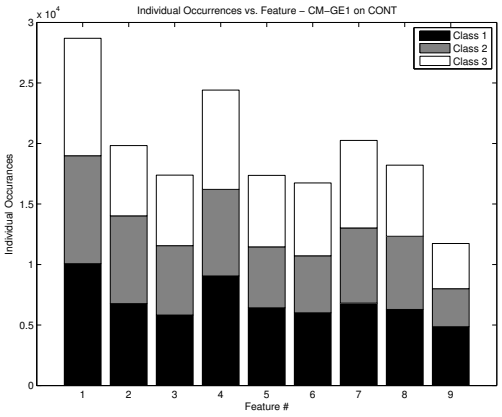
Figure E.2: Feature Analysis: CMGE 1, 2 on BUPA.



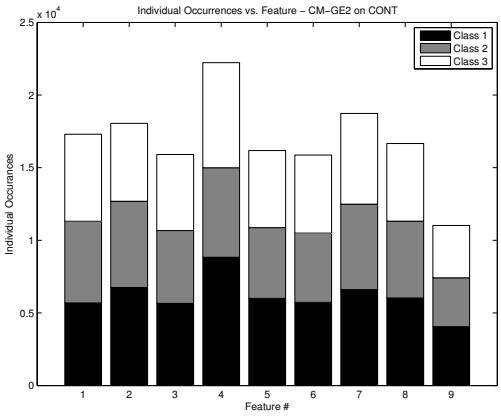
(a) CMGE1 Features per Individual



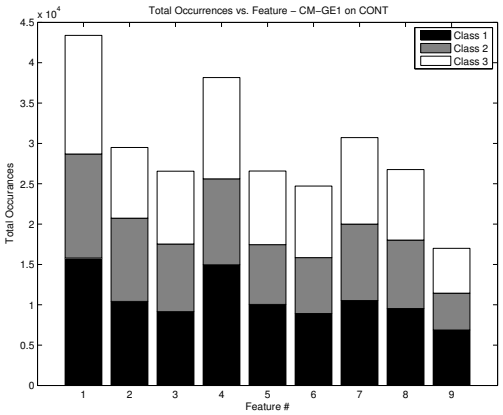
(b) CMGE2 Features per Individual



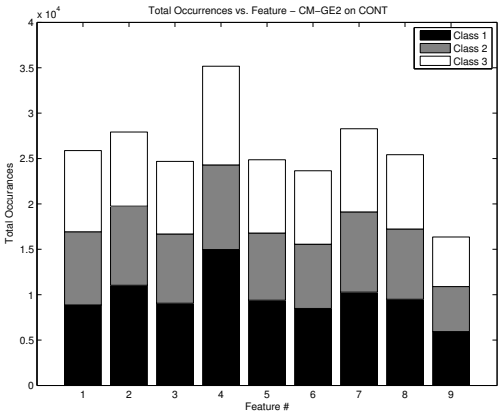
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

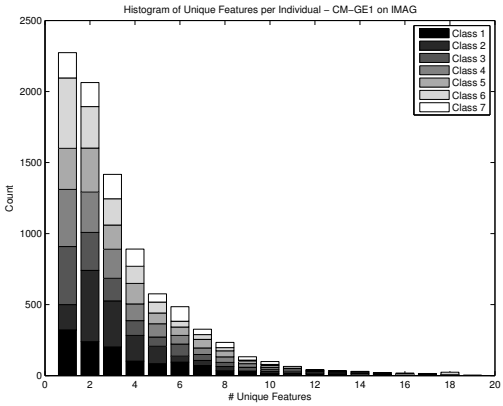


(e) CMGE1 Total Occurrences

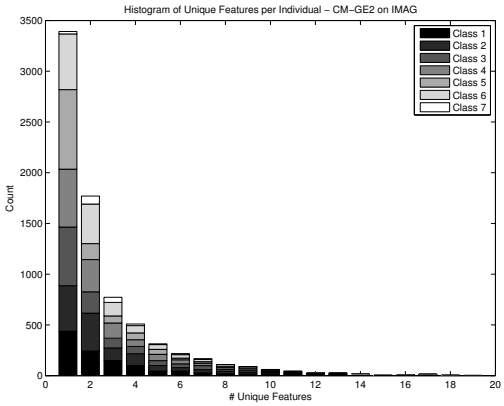


(f) CMGE2 Total Occurrences

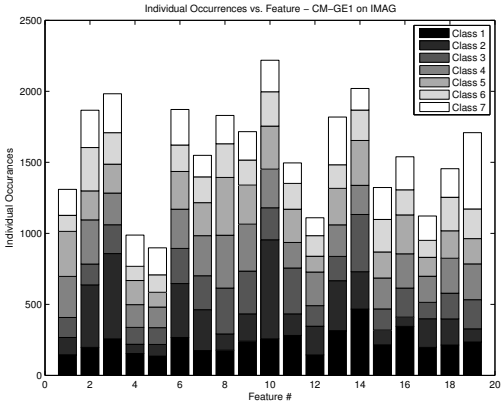
Figure E.3: Feature Analysis: CMGE 1, 2 on CONT.



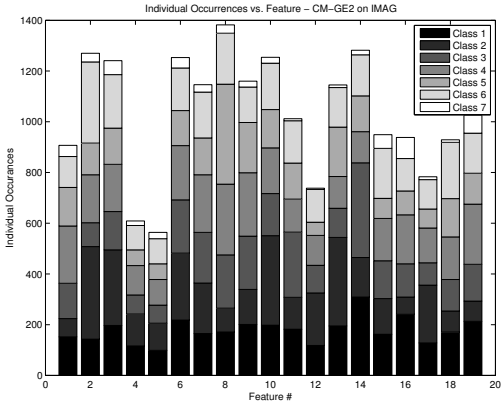
(a) CMGE1 Features per Individual



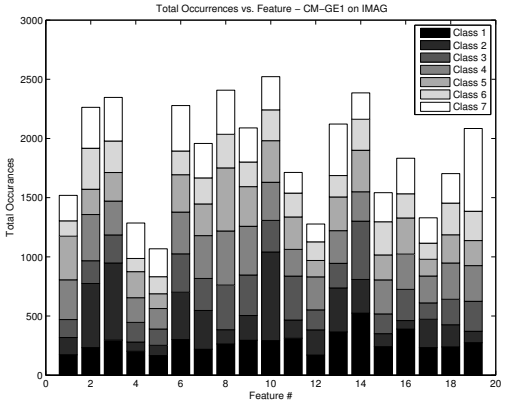
(b) CMGE2 Features per Individual



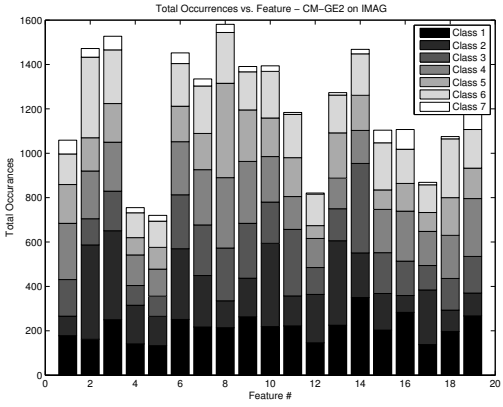
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

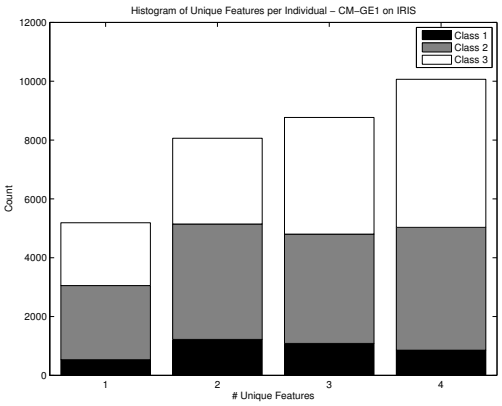


(e) CMGE1 Total Occurrences

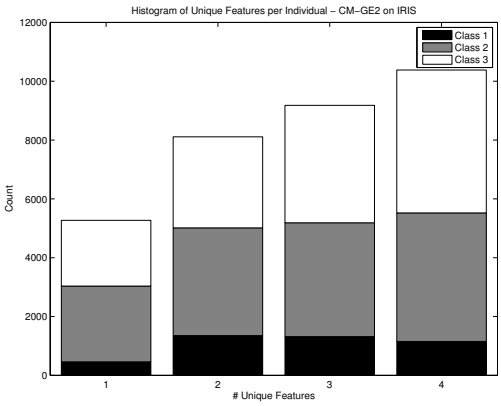


(f) CMGE2 Total Occurrences

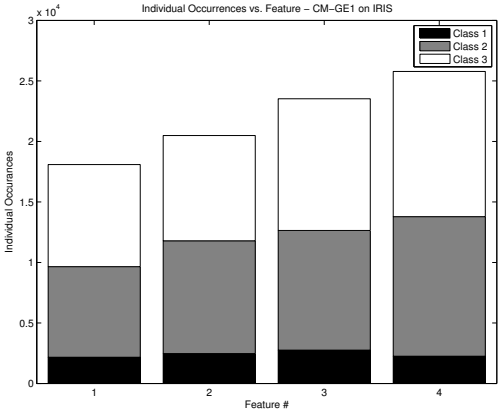
Figure E.4: Feature Analysis: CMGE 1, 2 on IMAG.



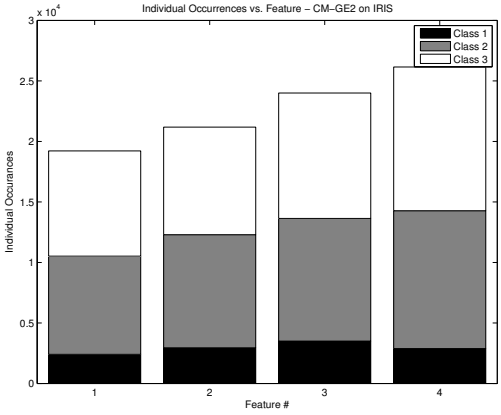
(a) CMGE1 Features per Individual



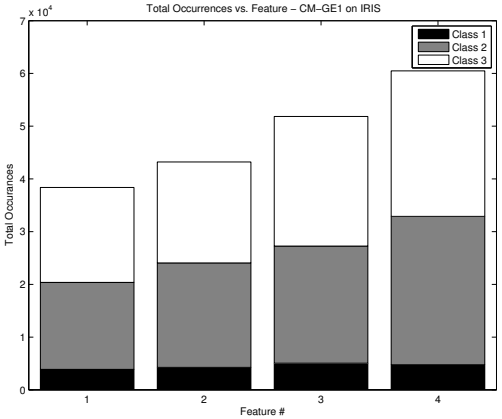
(b) CMGE2 Features per Individual



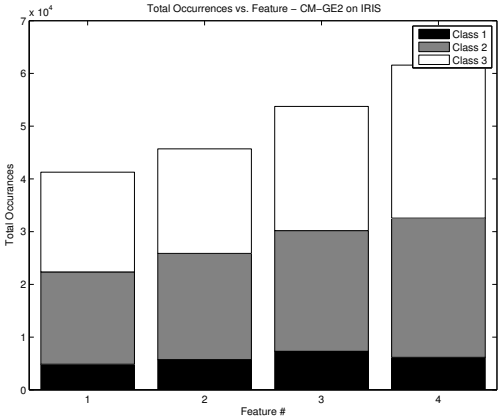
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

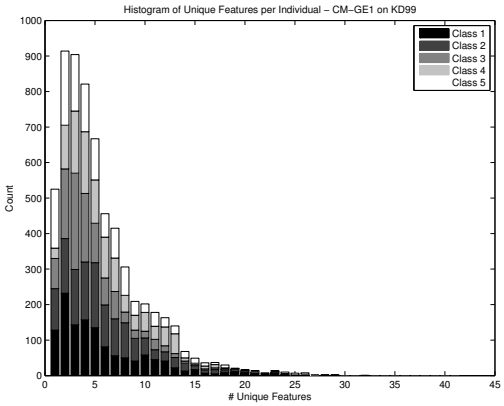


(e) CMGE1 Total Occurrences

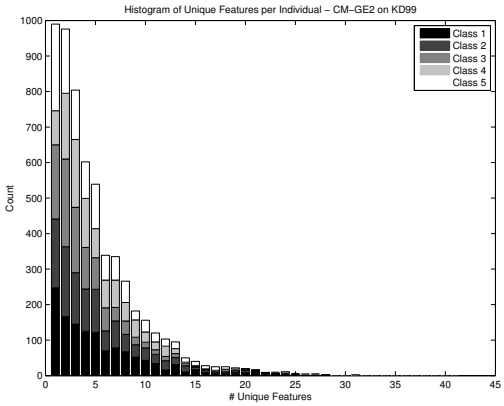


(f) CMGE2 Total Occurrences

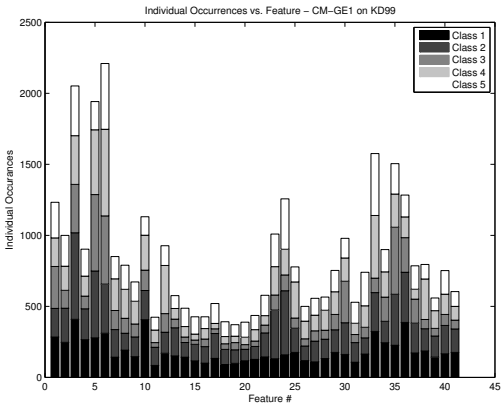
Figure E.5: Feature Analysis: CMGE 1, 2 on IRIS.



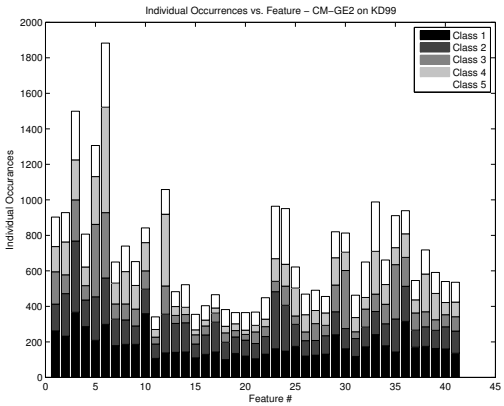
(a) CMGE1 Features per Individual



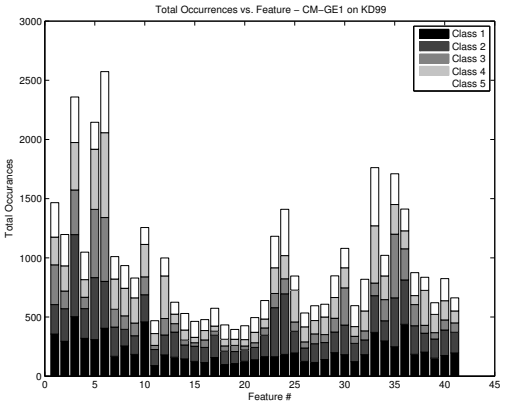
(b) CMGE2 Features per Individual



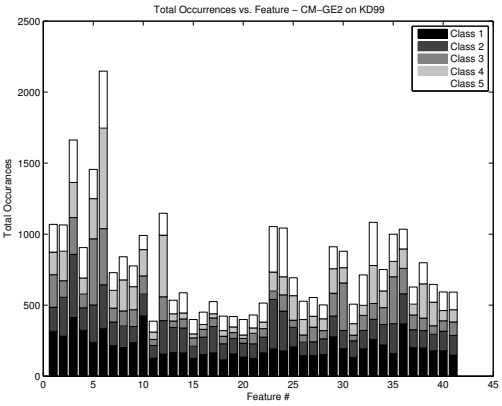
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

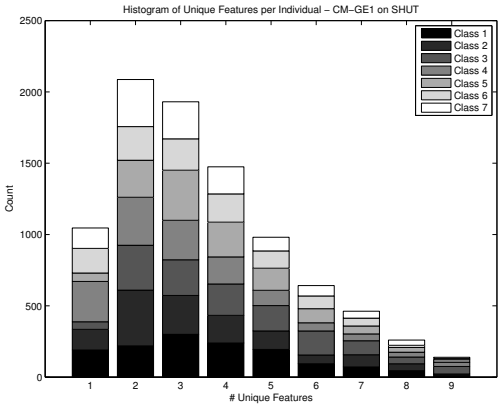


(e) CMGE1 Total Occurrences

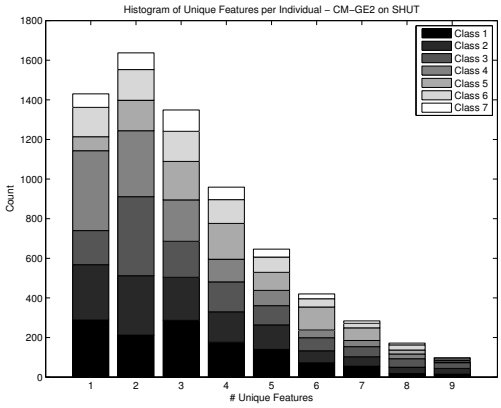


(f) CMGE2 Total Occurrences

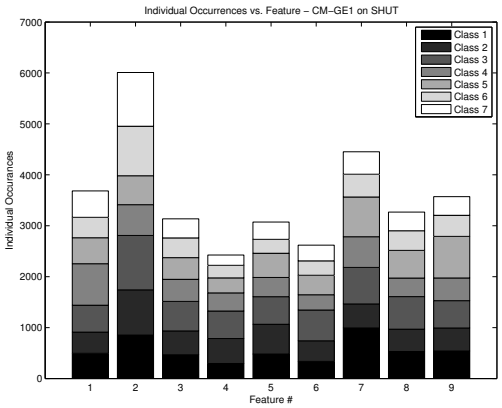
Figure E.6: Feature Analysis: CMGE 1, 2 on KD99.



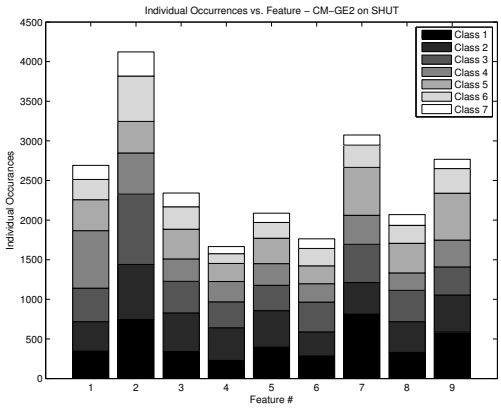
(a) CMGE1 Features per Individual



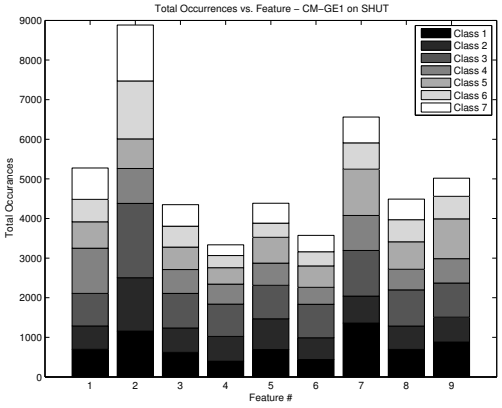
(b) CMGE2 Features per Individual



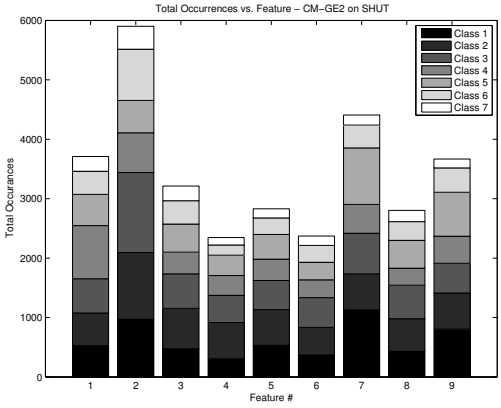
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

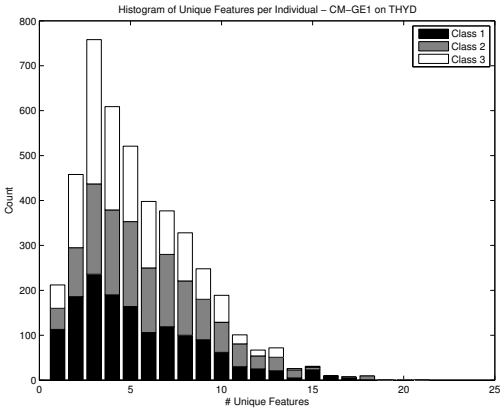


(e) CMGE1 Total Occurrences

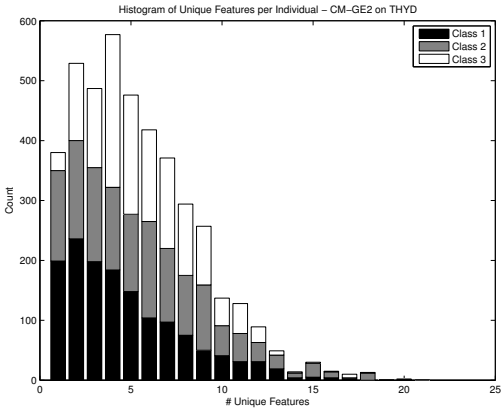


(f) CMGE2 Total Occurrences

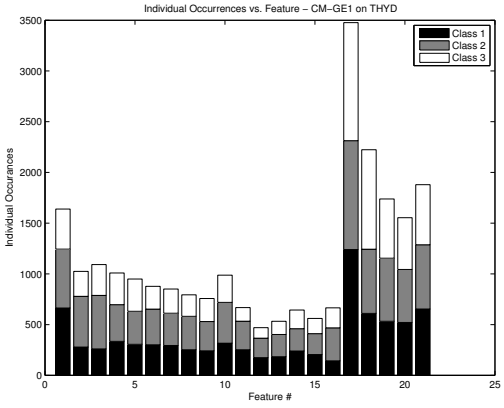
Figure E.7: Feature Analysis: CMGE 1, 2 on SHUT.



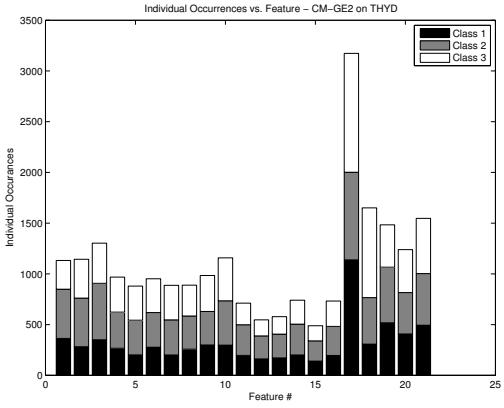
(a) CMGE1 Features per Individual



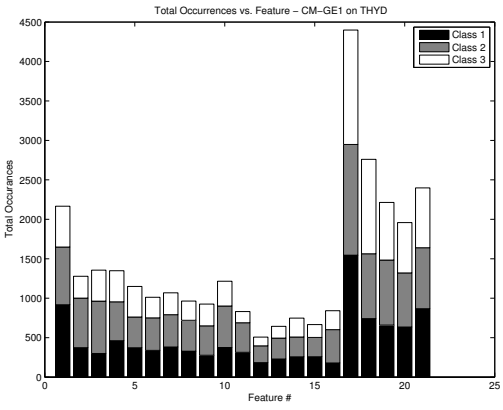
(b) CMGE2 Features per Individual



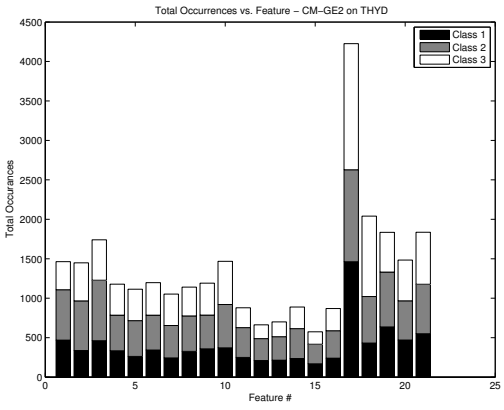
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

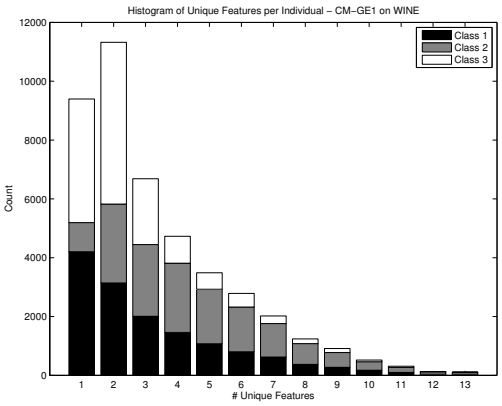


(e) CMGE1 Total Occurrences

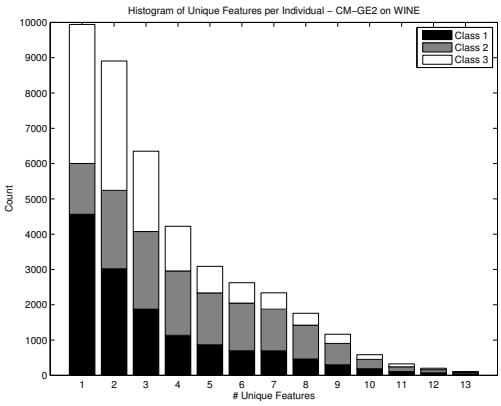


(f) CMGE2 Total Occurrences

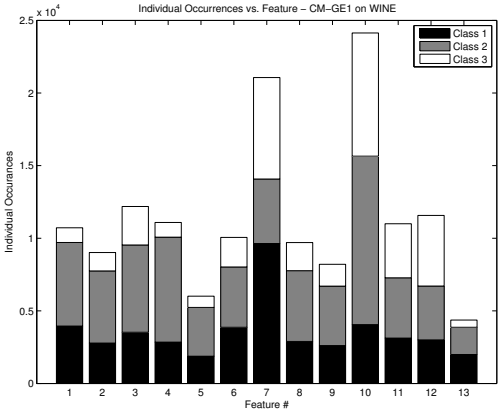
Figure E.8: Feature Analysis: CMGE 1, 2 on THYD.



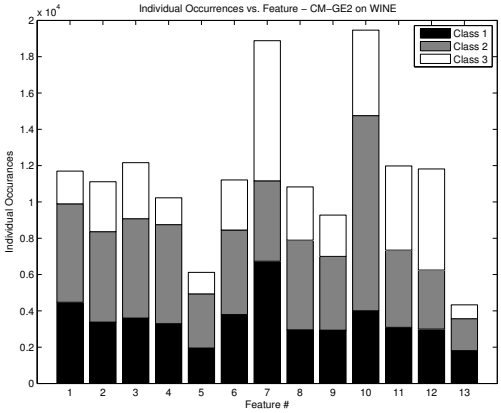
(a) CMGE1 Features per Individual



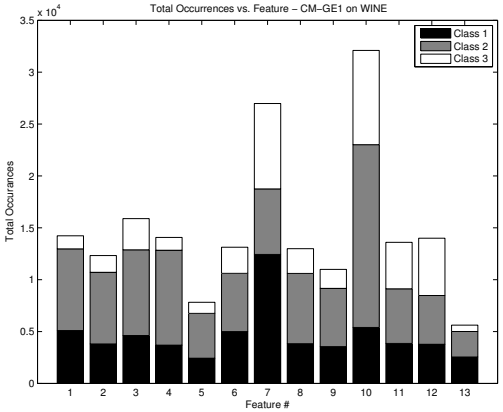
(b) CMGE2 Features per Individual



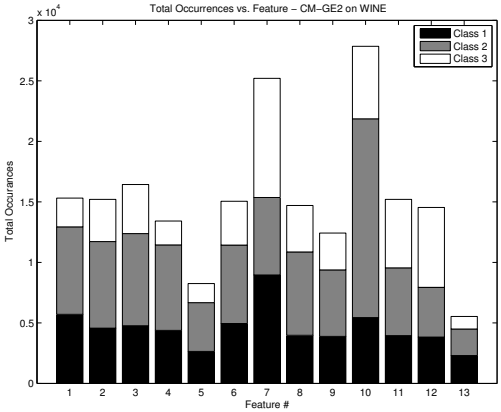
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences

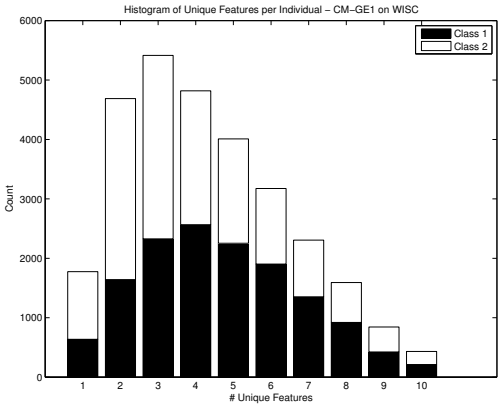


(e) CMGE1 Total Occurrences

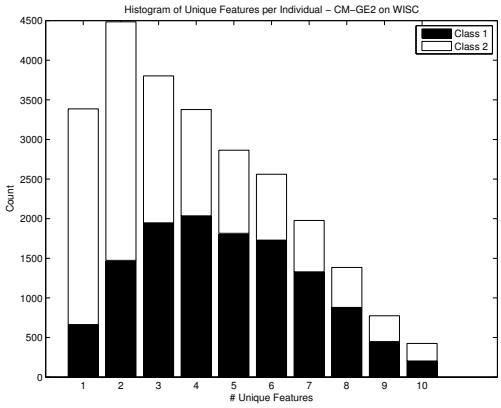


(f) CMGE2 Total Occurrences

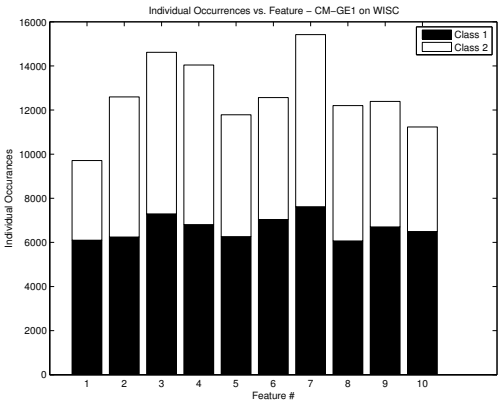
Figure E.9: Feature Analysis: CMGE 1, 2 on WINE.



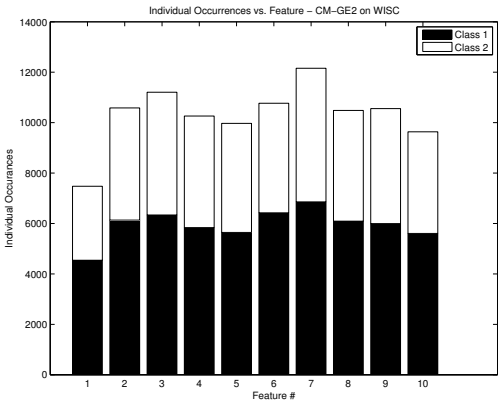
(a) CMGE1 Features per Individual



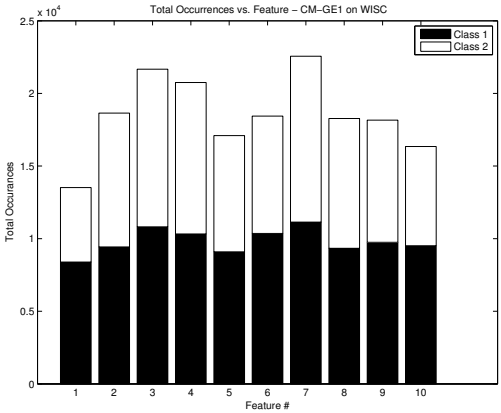
(b) CMGE2 Features per Individual



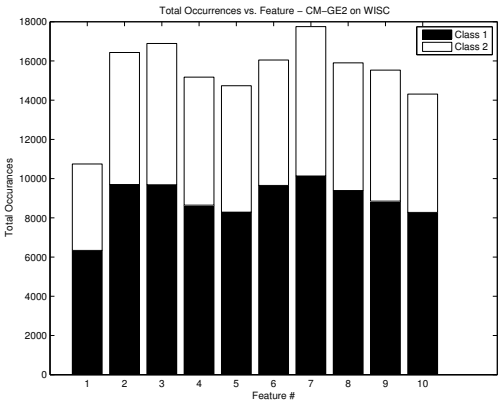
(c) CMGE1 Indiv. Occurrences



(d) CMGE2 Indiv. Occurrences



(e) CMGE1 Total Occurrences

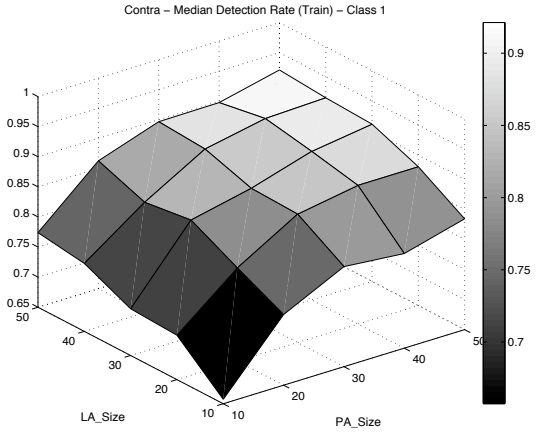


(f) CMGE2 Total Occurrences

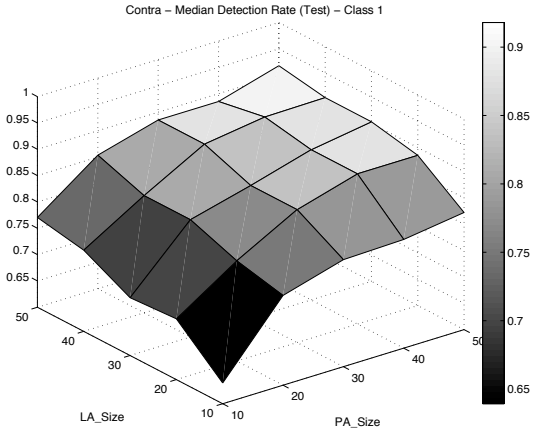
Figure E.10: Feature Analysis: CMGE 1, 2 on WISC.

Appendix F

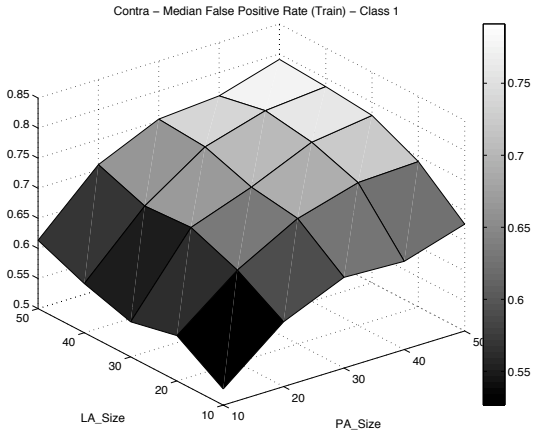
Results of (Class-wise) Surface Plots



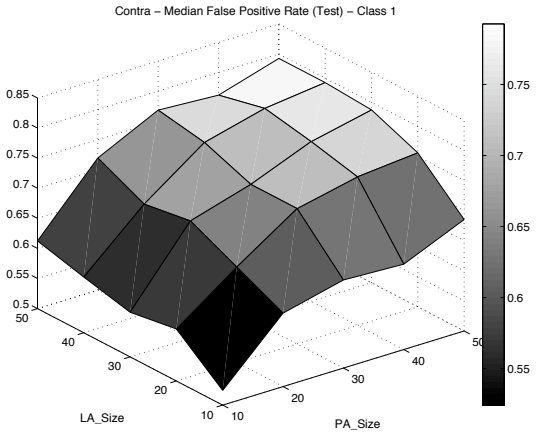
(a) Median DR (Class 1 - Train)



(b) Median DR (Class 1 - Test)

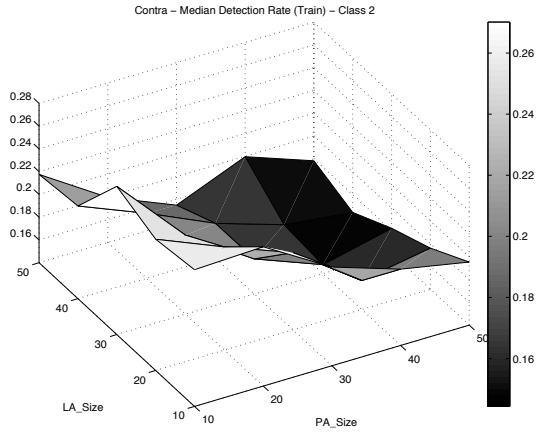


(c) Median FPR (Class 1 - Train)

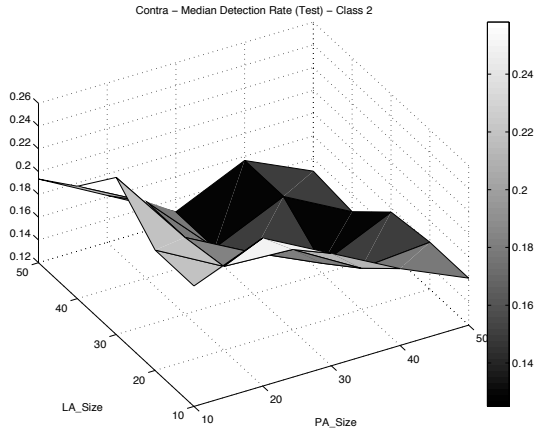


(d) Median FPR (Class 1 - Test)

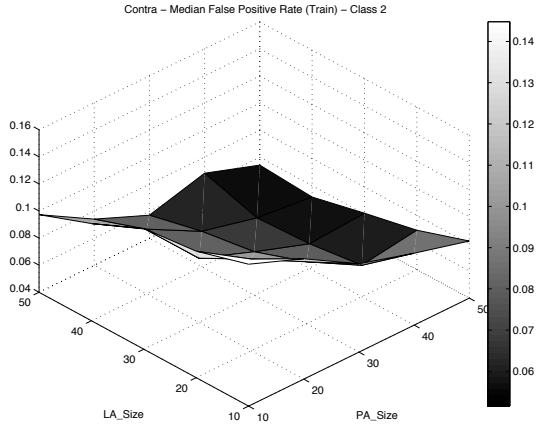
Figure F.1: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on CONT, class 1.



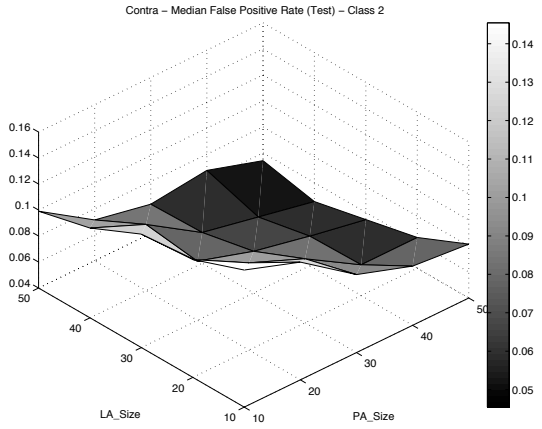
(a) Median DR (Class 2 - Train)



(b) Median DR (Class 2 - Test)

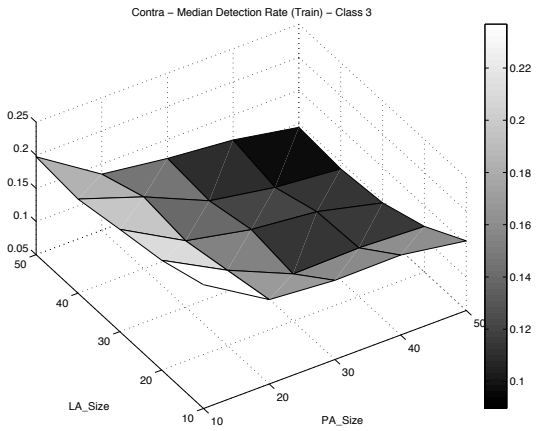


(c) Median FPR (Class 2 - Train)

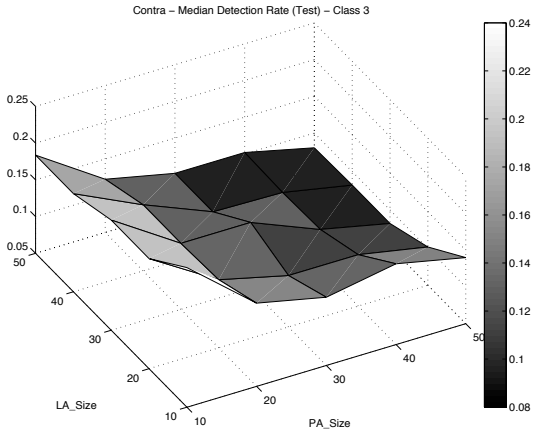


(d) Median FPR (Class 2 - Test)

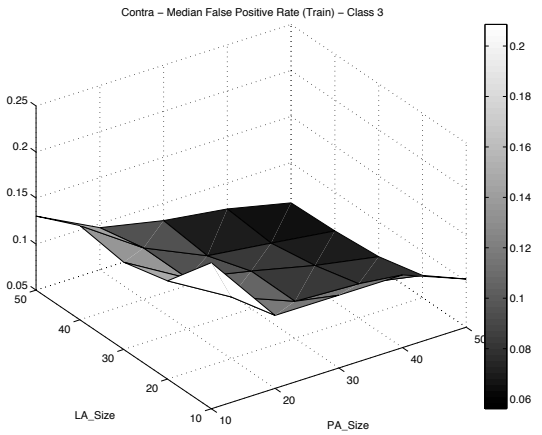
Figure F.2: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on CONT, class 2.



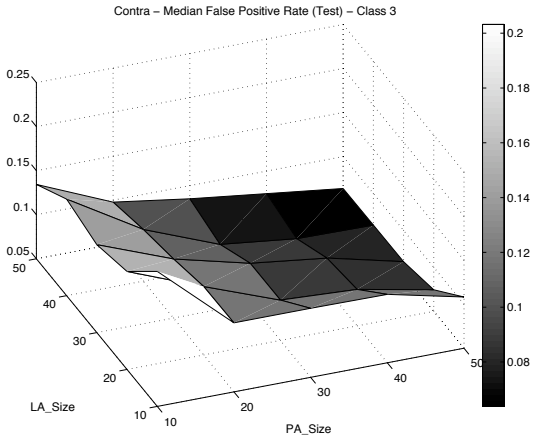
(a) Median DR (Class 3 - Train)



(b) Median DR (Class 3 - Test)

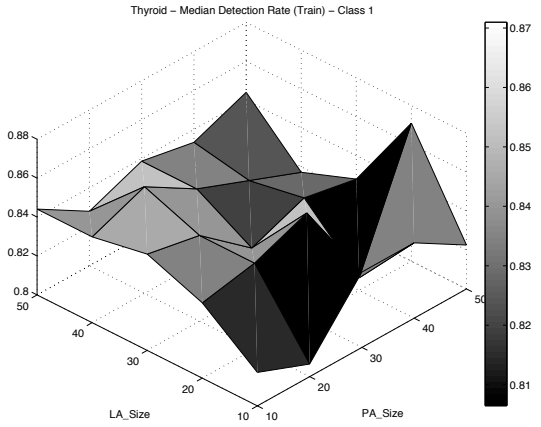


(c) Median FPR (Class 3 - Train)

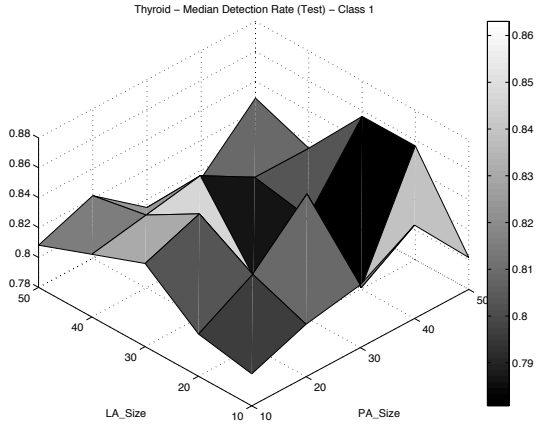


(d) Median FPR (Class 3 - Test)

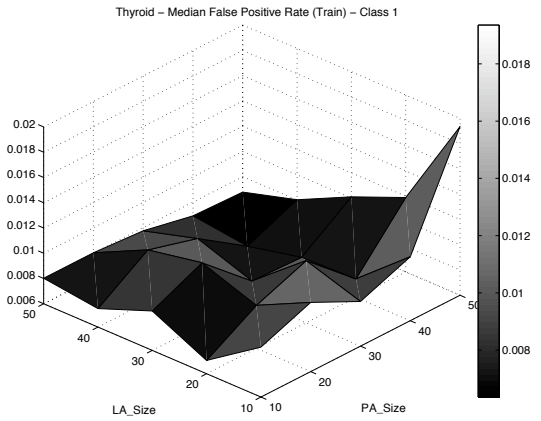
Figure F.3: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on CONT, class 3.



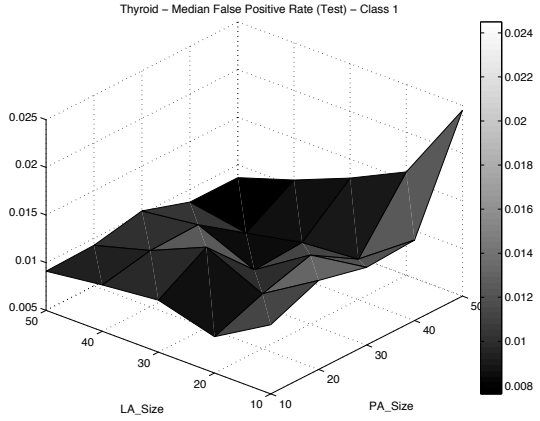
(a) Median DR (Class 1 - Train)



(b) Median DR (Class 1 - Test)

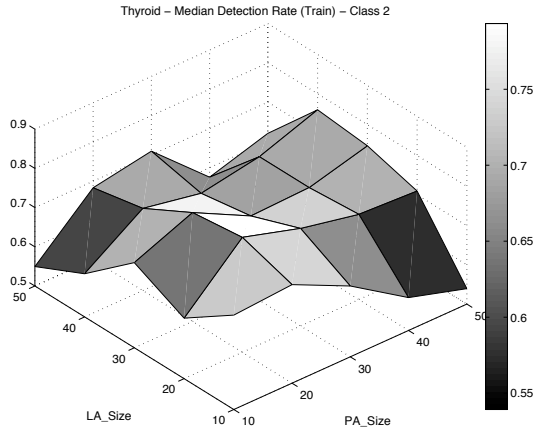


(c) Median FPR (Class 1 - Train)

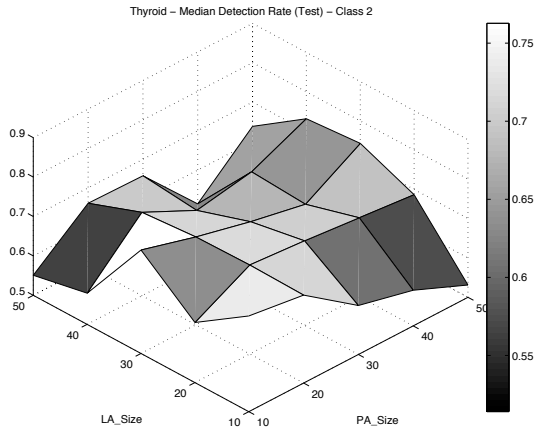


(d) Median FPR (Class 1 - Test)

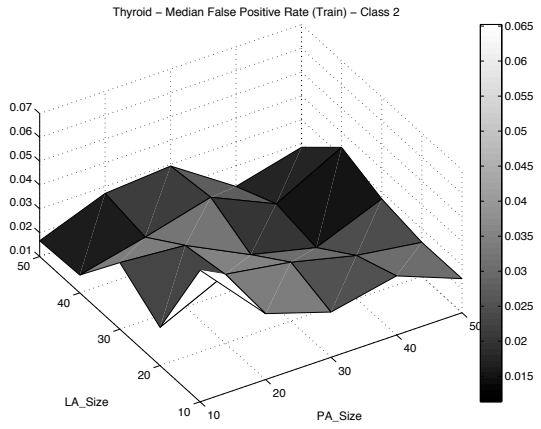
Figure F.4: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on THYD, class 1.



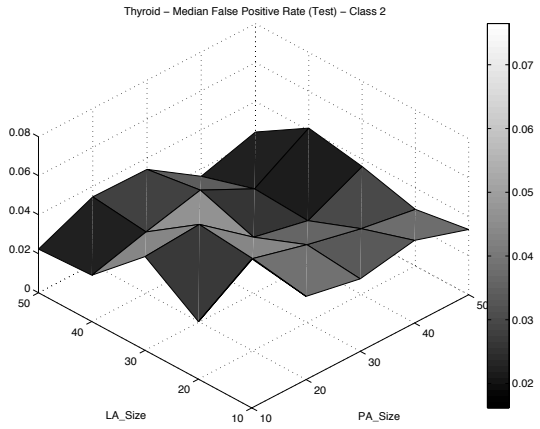
(a) Median DR (Class 2 - Train)



(b) Median DR (Class 2 - Test)

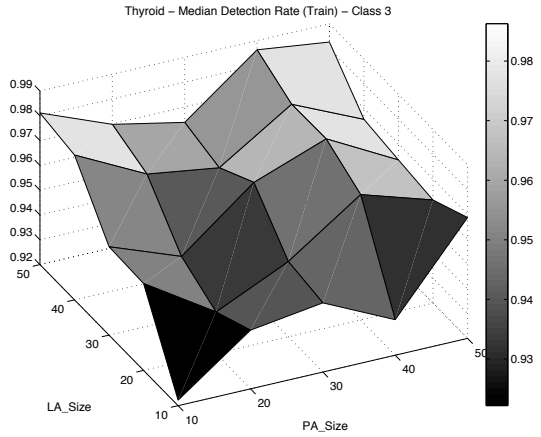


(c) Median FPR (Class 2 - Train)

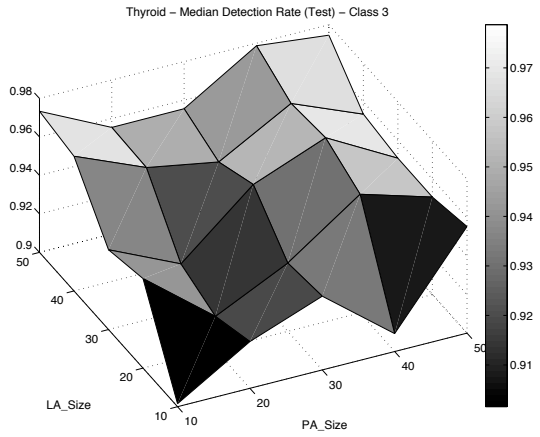


(d) Median FPR (Class 2 - Test)

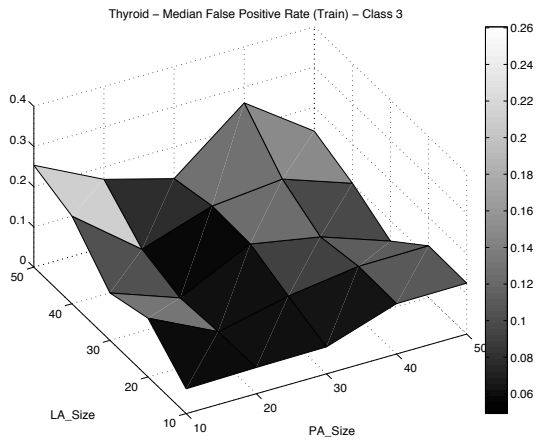
Figure F.5: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on THYD, class 2.



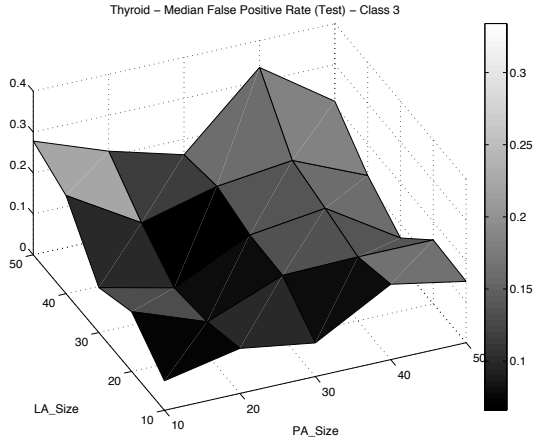
(a) Median DR (Class 3 - Train)



(b) Median DR (Class 3 - Test)



(c) Median FPR (Class 3 - Train)



(d) Median FPR (Class 3 - Test)

Figure F.6: Parameter analysis of CMGE 1 Detection (a) (b) and False Positive Rate (c) (d) on THYD, class 3.

Appendix G

Results: Quartile Summaries

The following tables present median (MED) along with first and third quartile (Q1, Q3 respectively) results for all GE, Deterministic and ANN experiments carried out in this work. Where applicable, results include overall accuracy, score, training time, solution size, number of solution participants, class-wise detection rate, and class-wise false positive rate. All quartiles are calculated over 50 independent initializations of the stochastic systems (GE, ANN). Where a train / test partitioning was not defined by the problem, ten-fold cross validation was performed for a total of 500 initializations. Deterministic experiments were not subject to multiple initializations; however, where ten-fold cross validation was employed, quartiles are reported across partitions.

G.1 Canonical GP Classifier

Table G.1: CAN-GE - CENS Quartile Summary

Overall Score (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.500 / 0.500	0.500 / 0.500	0.531 / 0.531
Overall Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.937 / 0.937	0.937 / 0.937	0.940 / 0.941
Training time (s)			
C	**Q1**	**MED**	**Q3**
-	139020.1	172408.7	269413.7
Classwise Soln Size			
C	**Q1**	**MED**	**Q3**
1	12.0	27.5	78.5
2	13.0	41.5	67.5
Classwise Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.937 / 0.937	0.937 / 0.937	0.940 / 0.941
2	0.937 / 0.937	0.937 / 0.937	0.940 / 0.941
Classwise Det Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.998 / 0.998	1.000 / 1.000	1.000 / 1.000
2	0.000 / 0.000	0.000 / 0.000	0.063 / 0.063
Classwise FP Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.937 / 0.937	1.000 / 1.000	1.000 / 1.000
2	0.000 / 0.000	0.000 / 0.000	0.002 / 0.002

Table G.2: CAN-GE - THYD Quartile Summary

Overall Score (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.441 / 0.438	0.625 / 0.610	0.718 / 0.706
Overall Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
-	0.933 / 0.933	0.944 / 0.942	0.957 / 0.953
Training time (s)			
C	**Q1**	**MED**	**Q3**
-	5241.0	7148.2	8529.2
Classwise Soln Size			
C	**Q1**	**MED**	**Q3**
1	25.0	54.5	112.0
2	29.0	72.0	111.0
3	36.0	67.0	116.0
Classwise Accuracy (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.980 / 0.983	0.989 / 0.989	0.991 / 0.990
2	0.949 / 0.948	0.950 / 0.949	0.960 / 0.958
3	0.939 / 0.940	0.951 / 0.947	0.969 / 0.960
Classwise Det Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.183 / 0.247	0.715 / 0.678	0.806 / 0.795
2	0.000 / 0.000	0.102 / 0.082	0.524 / 0.469
3	0.996 / 0.992	0.999 / 0.996	0.999 / 0.997
Classwise FP Rate (Train / Test)			
C	**Q1**	**MED**	**Q3**
1	0.000 / 0.001	0.002 / 0.003	0.004 / 0.006
2	0.000 / 0.000	0.002 / 0.004	0.009 / 0.011
3	0.349 / 0.408	0.623 / 0.674	0.785 / 0.788

G.2 Baseline GP Classifiers

Table G.3: Classwise Quartile Results - STD-GE

-		Soln Size			Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3	Q1	MED	Q3
BOST	1	13.5	28.0	49.5	0.633 / 0.575	0.713 / 0.706	0.801 / 0.824	0.098 / 0.088	0.146 / 0.143	0.202 / 0.235
	2	17.0	30.5	53.5	0.288 / 0.278	0.494 / 0.471	0.628 / 0.611	0.120 / 0.106	0.202 / 0.209	0.277 / 0.324
	3	12.5	21.0	43.0	0.624 / 0.588	0.729 / 0.706	0.819 / 0.824	0.069 / 0.061	0.153 / 0.147	0.271 / 0.279
BUPA	1	19.0	32.0	61.0	0.109 / 0.071	0.332 / 0.286	0.500 / 0.500	0.017 / 0.000	0.095 / 0.100	0.196 / 0.250
	2	8.0	12.0	26.0	0.804 / 0.750	0.905 / 0.900	0.983 / 1.000	0.500 / 0.500	0.668 / 0.714	0.891 / 0.929
CENS	1	15.0	38.5	70.0	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000
	2	15.0	23.5	56.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
CONT	1	12.0	21.0	38.0	0.638 / 0.613	0.767 / 0.754	0.907 / 0.903	0.456 / 0.463	0.633 / 0.622	0.911 / 0.907
	2	13.0	23.5	42.0	0.000 / 0.000	0.111 / 0.094	0.278 / 0.258	0.000 / 0.000	0.048 / 0.045	0.124 / 0.117
	3	15.0	25.0	50.0	0.049 / 0.040	0.212 / 0.204	0.388 / 0.380	0.050 / 0.054	0.163 / 0.188	0.325 / 0.330
IMAG	1	4.0	10.0	21.0	0.767 / 0.673	0.800 / 0.741	0.900 / 0.852	0.006 / 0.009	0.022 / 0.024	0.033 / 0.034
	2	13.0	21.0	62.0	0.967 / 0.970	1.000 / 0.982	1.000 / 0.993	0.006 / 0.017	0.019 / 0.020	0.033 / 0.035
	3	13.0	20.0	40.0	0.733 / 0.736	0.833 / 0.803	0.900 / 0.860	0.028 / 0.037	0.044 / 0.056	0.056 / 0.065
	4	23.0	35.0	54.0	0.033 / 0.037	0.267 / 0.313	0.633 / 0.563	0.000 / 0.006	0.039 / 0.049	0.122 / 0.129
	5	16.0	34.5	63.0	0.333 / 0.208	0.450 / 0.393	0.600 / 0.557	0.000 / 0.007	0.028 / 0.032	0.106 / 0.096
	6	16.0	32.5	67.0	0.100 / 0.027	0.750 / 0.695	0.967 / 0.942	0.000 / 0.003	0.056 / 0.067	0.117 / 0.135
	7	9.0	10.0	32.0	1.000 / 0.990	1.000 / 0.990	1.000 / 0.990	0.000 / 0.000	0.000 / 0.001	0.000 / 0.002
IRIS	1	22.0	42.0	78.0	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	25.0	37.5	73.5	0.889 / 0.800	0.933 / 1.000	0.956 / 1.000	0.000 / 0.000	0.023 / 0.000	0.034 / 0.100
	3	18.0	33.0	63.5	0.932 / 0.800	0.955 / 1.000	1.000 / 1.000	0.022 / 0.000	0.034 / 0.000	0.056 / 0.100
KD99	1	9.0	24.0	51.0	0.942 / 0.977	0.952 / 0.991	0.985 / 0.997	0.071 / 0.198	0.082 / 0.254	0.171 / 0.379
	2	16.0	30.0	67.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.001	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	13.0	29.0	51.0	0.811 / 0.622	0.934 / 0.852	0.951 / 0.886	0.010 / 0.013	0.028 / 0.032	0.066 / 0.043
	4	16.0	28.5	57.0	0.000 / 0.000	0.000 / 0.000	0.002 / 0.010	0.000 / 0.000	0.000 / 0.000	0.001 / 0.004
	5	16.0	29.0	78.0	0.000 / 0.000	0.000 / 0.000	0.019 / 0.000	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
PIMA	1	12.0	24.0	41.0	0.956 / 0.940	0.987 / 0.980	1.000 / 1.000	0.859 / 0.852	0.963 / 0.963	0.996 / 1.000
	2	16.0	25.0	51.5	0.004 / 0.000	0.037 / 0.037	0.141 / 0.148	0.000 / 0.000	0.013 / 0.020	0.044 / 0.060
SHUT	1	8.0	17.0	32.0	0.990 / 0.989	0.995 / 0.995	0.999 / 0.999	0.645 / 0.635	0.761 / 0.747	0.923 / 0.922
	2	17.0	36.0	47.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	13.0	24.0	49.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	4	21.0	30.5	51.0	0.000 / 0.000	0.000 / 0.000	0.021 / 0.018	0.000 / 0.000	0.000 / 0.000	0.007 / 0.008
	5	8.0	22.0	49.0	0.006 / 0.005	0.174 / 0.163	0.567 / 0.582	0.000 / 0.000	0.001 / 0.001	0.007 / 0.005
	6	13.0	21.0	42.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	13.0	25.0	42.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
THYD	1	30.0	47.5	82.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	37.0	55.5	98.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	37.0	57.5	91.0	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.996 / 1.000	1.000 / 1.000	1.000 / 1.000
WINE	1	16.0	26.0	50.0	0.868 / 0.833	0.925 / 1.000	0.963 / 1.000	0.047 / 0.000	0.075 / 0.083	0.131 / 0.167
	2	16.0	26.0	47.5	0.734 / 0.714	0.828 / 0.857	0.891 / 0.857	0.021 / 0.000	0.042 / 0.000	0.094 / 0.091
	3	14.5	27.0	56.0	0.953 / 1.000	1.000 / 1.000	1.000 / 1.000	0.008 / 0.000	0.017 / 0.000	0.034 / 0.077
WISC	1	23.0	40.0	70.0	0.939 / 0.932	0.959 / 0.955	0.967 / 0.977	0.028 / 0.042	0.052 / 0.043	0.099 / 0.130
	2	16.0	36.0	65.0	0.901 / 0.870	0.948 / 0.957	0.972 / 0.958	0.033 / 0.023	0.041 / 0.045	0.061 / 0.068

Table G.4: Classwise Quartile Results - RSS-GE

-		Soln Size			Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3	Q1	MED	Q3
BOST	1	11.0	19.0	41.5	0.647 / 0.588	0.748 / 0.765	0.827 / 0.824	0.128 / 0.114	0.174 / 0.182	0.230 / 0.265
	2	21.0	35.0	65.0	0.397 / 0.353	0.506 / 0.471	0.603 / 0.611	0.164 / 0.125	0.219 / 0.219	0.279 / 0.312
	3	11.0	22.5	46.0	0.559 / 0.529	0.658 / 0.647	0.765 / 0.812	0.078 / 0.060	0.141 / 0.143	0.183 / 0.206
BUPA	1	15.5	25.0	53.0	0.569 / 0.500	0.648 / 0.571	0.688 / 0.714	0.335 / 0.300	0.397 / 0.421	0.474 / 0.500
	2	16.0	30.0	70.5	0.526 / 0.500	0.603 / 0.579	0.665 / 0.700	0.312 / 0.286	0.352 / 0.429	0.431 / 0.500
CENS	1	17.0	33.0	66.0	0.578 / 0.581	0.675 / 0.677	0.744 / 0.745	0.114 / 0.113	0.171 / 0.173	0.259 / 0.261
	2	10.0	15.5	30.0	0.741 / 0.739	0.829 / 0.827	0.886 / 0.887	0.256 / 0.255	0.325 / 0.323	0.422 / 0.419
CONT	1	15.0	28.0	54.0	0.392 / 0.371	0.456 / 0.443	0.527 / 0.532	0.195 / 0.188	0.255 / 0.263	0.316 / 0.338
	2	16.0	29.0	48.5	0.415 / 0.406	0.518 / 0.500	0.611 / 0.594	0.222 / 0.223	0.282 / 0.284	0.342 / 0.359
	3	20.0	38.0	70.0	0.298 / 0.265	0.380 / 0.367	0.453 / 0.460	0.222 / 0.217	0.284 / 0.287	0.342 / 0.357
IMAG	1	9.0	16.0	32.0	0.733 / 0.630	0.833 / 0.753	0.967 / 0.926	0.011 / 0.013	0.028 / 0.027	0.056 / 0.050
	2	9.0	26.5	48.0	0.933 / 0.910	1.000 / 0.977	1.000 / 0.997	0.011 / 0.018	0.028 / 0.025	0.028 / 0.031
	3	20.0	29.5	62.0	0.267 / 0.251	0.600 / 0.520	0.733 / 0.773	0.017 / 0.030	0.042 / 0.050	0.061 / 0.076
	4	16.0	26.5	68.0	0.300 / 0.210	0.467 / 0.395	0.567 / 0.487	0.039 / 0.046	0.061 / 0.069	0.106 / 0.109
	5	22.0	34.5	56.0	0.300 / 0.258	0.567 / 0.517	0.667 / 0.641	0.022 / 0.029	0.067 / 0.078	0.117 / 0.121
	6	18.0	31.0	59.0	0.667 / 0.565	0.800 / 0.784	0.900 / 0.870	0.044 / 0.052	0.086 / 0.084	0.128 / 0.145
	7	9.0	9.0	18.0	1.000 / 0.987	1.000 / 0.990	1.000 / 0.990	0.000 / 0.000	0.000 / 0.001	0.000 / 0.002
IRIS	1	23.0	44.0	81.0	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	24.0	40.5	74.5	0.889 / 0.800	0.933 / 1.000	0.956 / 1.000	0.000 / 0.000	0.023 / 0.000	0.057 / 0.100
	3	18.0	34.0	70.0	0.886 / 0.800	0.955 / 1.000	1.000 / 1.000	0.011 / 0.000	0.034 / 0.000	0.057 / 0.100
KD99	1	18.0	37.0	106.0	0.899 / 0.957	0.944 / 0.976	0.979 / 0.987	0.032 / 0.132	0.045 / 0.178	0.067 / 0.215
	2	13.0	25.0	44.0	0.001 / 0.001	0.053 / 0.022	0.361 / 0.156	0.001 / 0.005	0.007 / 0.008	0.027 / 0.028
	3	17.0	42.5	76.0	0.901 / 0.809	0.946 / 0.861	0.952 / 0.890	0.004 / 0.015	0.010 / 0.026	0.016 / 0.033
	4	14.0	29.5	56.0	0.063 / 0.043	0.169 / 0.168	0.440 / 0.436	0.005 / 0.004	0.010 / 0.010	0.034 / 0.021
	5	9.0	19.0	40.0	0.019 / 0.000	0.115 / 0.043	0.462 / 0.314	0.001 / 0.003	0.005 / 0.007	0.025 / 0.016
PIMA	1	21.0	35.0	65.0	0.564 / 0.540	0.653 / 0.640	0.727 / 0.720	0.357 / 0.358	0.432 / 0.444	0.506 / 0.556
	2	21.0	32.5	58.0	0.494 / 0.444	0.568 / 0.556	0.643 / 0.642	0.273 / 0.280	0.347 / 0.360	0.436 / 0.460
SHUT	1	17.0	29.5	55.0	0.758 / 0.759	0.803 / 0.805	0.923 / 0.920	0.003 / 0.004	0.052 / 0.052	0.188 / 0.172
	2	13.0	22.5	42.0	0.000 / 0.000	0.432 / 0.615	0.838 / 0.923	0.000 / 0.001	0.002 / 0.002	0.012 / 0.013
	3	17.0	27.0	42.0	0.295 / 0.333	0.750 / 0.731	0.970 / 0.974	0.002 / 0.002	0.005 / 0.006	0.035 / 0.035
	4	18.0	40.0	72.0	0.517 / 0.516	0.681 / 0.680	0.975 / 0.973	0.063 / 0.065	0.144 / 0.145	0.203 / 0.200
	5	13.0	23.0	42.0	0.000 / 0.000	0.552 / 0.542	0.903 / 0.897	0.000 / 0.000	0.000 / 0.000	0.017 / 0.016
	6	8.0	12.0	23.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	26.0	42.0	66.0	0.000 / 0.000	0.182 / 0.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
THYD	1	10.0	30.5	59.0	0.742 / 0.685	0.887 / 0.842	0.968 / 0.973	0.014 / 0.019	0.043 / 0.048	0.121 / 0.145
	2	10.0	19.0	32.0	0.084 / 0.090	0.545 / 0.593	0.775 / 0.785	0.041 / 0.048	0.127 / 0.160	0.540 / 0.539
	3	13.0	41.0	83.0	0.409 / 0.419	0.779 / 0.747	0.892 / 0.862	0.011 / 0.020	0.074 / 0.058	0.148 / 0.164
WINE	1	16.0	26.0	46.0	0.868 / 0.833	0.925 / 1.000	0.972 / 1.000	0.056 / 0.000	0.093 / 0.083	0.167 / 0.167
	2	17.0	30.0	55.5	0.656 / 0.571	0.797 / 0.714	0.859 / 0.857	0.021 / 0.000	0.042 / 0.000	0.082 / 0.091
	3	14.0	27.0	53.0	0.943 / 1.000	0.977 / 1.000	1.000 / 1.000	0.017 / 0.000	0.034 / 0.000	0.060 / 0.077
WISC	1	23.0	39.0	71.0	0.914 / 0.907	0.944 / 0.932	0.960 / 0.977	0.019 / 0.000	0.042 / 0.042	0.080 / 0.087
	2	18.0	43.0	83.5	0.920 / 0.913	0.958 / 0.958	0.981 / 1.000	0.040 / 0.023	0.056 / 0.068	0.086 / 0.093

G.3 Scalable GP Classifiers

Table G.5: Classwise Quartile Results - CM-GE1

-		Soln Size / Participants			Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3	Q1	MED	Q3
BOST	1	14.0 / 30.0	24.0 / 30.0	42.0 / 30.0	0.827 / 0.765	0.860 / 0.875	0.887 / 0.938	0.187 / 0.182	0.217 / 0.229	0.248 / 0.273
	2	13.0 / 30.0	18.0 / 30.0	31.0 / 30.0	0.327 / 0.278	0.404 / 0.353	0.481 / 0.471	0.070 / 0.059	0.093 / 0.094	0.117 / 0.147
	3	16.5 / 30.0	34.5 / 30.0	59.0 / 30.0	0.793 / 0.750	0.827 / 0.824	0.859 / 0.882	0.124 / 0.114	0.151 / 0.152	0.183 / 0.212
BUPA	1	26.0 / 30.0	38.5 / 30.0	54.0 / 30.0	0.215 / 0.143	0.361 / 0.286	0.469 / 0.429	0.050 / 0.050	0.101 / 0.100	0.151 / 0.200
	2	29.0 / 30.0	43.0 / 30.0	59.8 / 30.0	0.849 / 0.800	0.899 / 0.900	0.950 / 0.950	0.531 / 0.571	0.639 / 0.714	0.785 / 0.857
CENS	1	41.0 / 29.0	57.2 / 29.0	75.5 / 30.0	0.963 / 0.964	0.980 / 0.980	0.993 / 0.994	0.696 / 0.699	0.807 / 0.802	0.920 / 0.925
	2	24.0 / 30.0	33.8 / 30.0	58.0 / 30.0	0.080 / 0.075	0.193 / 0.198	0.304 / 0.301	0.007 / 0.006	0.020 / 0.020	0.037 / 0.036
CONT	1	26.0 / 27.0	40.0 / 28.0	63.8 / 29.0	0.657 / 0.645	0.742 / 0.738	0.817 / 0.806	0.435 / 0.439	0.520 / 0.537	0.614 / 0.646
	2	18.2 / 28.0	25.2 / 29.0	35.0 / 30.0	0.244 / 0.219	0.342 / 0.312	0.421 / 0.406	0.100 / 0.100	0.143 / 0.145	0.182 / 0.197
	3	22.0 / 29.0	29.8 / 29.0	42.8 / 30.0	0.182 / 0.143	0.251 / 0.240	0.332 / 0.327	0.104 / 0.102	0.152 / 0.163	0.218 / 0.234
IMAG	1	16.0 / 25.0	20.8 / 27.0	27.0 / 28.0	0.900 / 0.774	0.933 / 0.896	0.967 / 0.956	0.033 / 0.034	0.050 / 0.058	0.072 / 0.081
	2	14.0 / 29.0	17.8 / 30.0	26.5 / 30.0	1.000 / 0.990	1.000 / 0.997	1.000 / 1.000	0.017 / 0.024	0.025 / 0.034	0.039 / 0.043
	3	13.0 / 24.0	15.0 / 27.0	25.0 / 28.0	0.800 / 0.749	0.867 / 0.806	0.933 / 0.849	0.028 / 0.044	0.039 / 0.054	0.050 / 0.071
	4	14.0 / 26.0	18.2 / 28.0	30.0 / 29.0	0.300 / 0.107	0.433 / 0.198	0.567 / 0.313	0.000 / 0.010	0.011 / 0.017	0.017 / 0.026
	5	15.0 / 25.0	21.5 / 27.0	32.0 / 28.0	0.400 / 0.262	0.533 / 0.366	0.667 / 0.456	0.006 / 0.013	0.011 / 0.026	0.022 / 0.040
	6	13.5 / 26.0	16.5 / 28.0	20.0 / 29.0	0.900 / 0.860	0.950 / 0.911	0.967 / 0.962	0.050 / 0.088	0.067 / 0.096	0.094 / 0.125
	7	24.0 / 16.0	32.5 / 23.0	52.0 / 26.0	1.000 / 0.983	1.000 / 0.990	1.000 / 0.990	0.000 / 0.001	0.006 / 0.007	0.028 / 0.023
IRIS	1	19.2 / 2.0	27.0 / 3.0	42.0 / 8.0	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	21.5 / 30.0	29.0 / 30.0	45.0 / 30.0	0.933 / 0.800	0.933 / 1.000	0.956 / 1.000	0.000 / 0.000	0.011 / 0.000	0.023 / 0.000
	3	24.8 / 29.0	36.0 / 30.0	55.0 / 30.0	0.955 / 1.000	0.977 / 1.000	1.000 / 1.000	0.023 / 0.000	0.034 / 0.000	0.034 / 0.100
KD99	1	25.0 / 26.0	38.8 / 28.0	60.5 / 30.0	0.922 / 0.972	0.980 / 0.988	0.992 / 0.994	0.048 / 0.196	0.064 / 0.225	0.072 / 0.241
	2	30.0 / 27.0	40.8 / 28.5	52.0 / 30.0	0.192 / 0.059	0.511 / 0.112	0.584 / 0.155	0.001 / 0.001	0.002 / 0.002	0.006 / 0.007
	3	21.0 / 21.0	26.5 / 24.0	37.0 / 28.0	0.948 / 0.853	0.951 / 0.859	0.956 / 0.862	0.004 / 0.009	0.009 / 0.015	0.019 / 0.026
	4	29.5 / 20.0	38.0 / 24.5	59.0 / 27.0	0.365 / 0.255	0.468 / 0.319	0.626 / 0.374	0.000 / 0.001	0.000 / 0.002	0.002 / 0.003
	5	25.0 / 23.0	29.0 / 26.0	47.5 / 28.0	0.192 / 0.100	0.490 / 0.314	0.692 / 0.500	0.001 / 0.002	0.004 / 0.007	0.017 / 0.019
PIMA	1	21.0 / 30.0	30.0 / 30.0	49.5 / 30.0	0.878 / 0.860	0.913 / 0.920	0.947 / 0.960	0.521 / 0.519	0.625 / 0.667	0.776 / 0.815
	2	21.0 / 30.0	31.0 / 30.0	46.0 / 30.0	0.224 / 0.185	0.375 / 0.333	0.479 / 0.481	0.053 / 0.040	0.087 / 0.080	0.122 / 0.140
SHUT	1	18.0 / 27.0	23.0 / 28.0	36.0 / 30.0	0.756 / 0.751	0.868 / 0.866	0.952 / 0.948	0.015 / 0.016	0.048 / 0.046	0.108 / 0.105
	2	21.5 / 26.0	26.0 / 27.5	33.5 / 29.0	0.622 / 0.462	0.797 / 0.654	0.919 / 0.846	0.000 / 0.000	0.002 / 0.002	0.007 / 0.006
	3	27.0 / 27.0	36.5 / 28.5	56.0 / 30.0	0.462 / 0.436	0.693 / 0.705	0.833 / 0.872	0.005 / 0.006	0.023 / 0.023	0.091 / 0.090
	4	18.0 / 27.0	22.0 / 28.0	29.0 / 29.0	0.828 / 0.833	0.922 / 0.923	0.966 / 0.969	0.004 / 0.003	0.011 / 0.012	0.022 / 0.023
	5	26.0 / 23.0	35.0 / 27.0	53.0 / 29.0	0.992 / 0.990	0.993 / 0.993	0.994 / 0.994	0.000 / 0.000	0.000 / 0.000	0.000 / 0.001
	6	22.0 / 22.0	30.8 / 24.5	51.5 / 27.0	0.833 / 0.750	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.001 / 0.001	0.006 / 0.006
	7	17.0 / 23.0	20.2 / 26.0	30.5 / 28.0	0.909 / 0.500	1.000 / 1.000	1.000 / 1.000	0.001 / 0.001	0.013 / 0.014	0.101 / 0.103
THYD	1	26.0 / 29.0	43.5 / 30.0	65.0 / 30.0	0.774 / 0.740	0.817 / 0.808	0.871 / 0.863	0.005 / 0.007	0.007 / 0.008	0.012 / 0.014
	2	35.0 / 29.0	49.2 / 30.0	73.0 / 30.0	0.618 / 0.599	0.801 / 0.802	0.885 / 0.898	0.021 / 0.029	0.047 / 0.059	0.073 / 0.089
	3	28.0 / 30.0	32.5 / 30.0	49.5 / 30.0	0.911 / 0.884	0.943 / 0.928	0.975 / 0.963	0.028 / 0.020	0.076 / 0.098	0.208 / 0.256
WINE	1	13.0 / 30.0	21.0 / 30.0	35.0 / 30.0	0.916 / 0.833	0.943 / 1.000	0.963 / 1.000	0.019 / 0.000	0.037 / 0.000	0.056 / 0.083
	2	26.0 / 30.0	43.0 / 30.0	61.0 / 30.0	0.875 / 0.857	0.906 / 0.857	0.938 / 1.000	0.021 / 0.000	0.031 / 0.000	0.052 / 0.091
	3	13.0 / 30.0	14.0 / 30.0	21.0 / 30.0	0.977 / 1.000	1.000 / 1.000	1.000 / 1.000	0.009 / 0.000	0.017 / 0.000	0.025 / 0.000
WISC	1	36.0 / 28.0	49.0 / 29.5	63.0 / 30.0	0.954 / 0.953	0.962 / 0.977	0.970 / 0.977	0.038 / 0.042	0.061 / 0.083	0.089 / 0.125
	2	29.0 / 30.0	37.0 / 30.0	48.0 / 30.0	0.911 / 0.875	0.939 / 0.917	0.962 / 0.958	0.030 / 0.023	0.038 / 0.023	0.046 / 0.047

Table G.6: Classwise Quartile Results - CM-GE2

-		Soln Size / Participants			Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3	Q1	MED	Q3
BOST	1	14.0 / 27.0	22.0 / 29.0	44.0 / 30.0	0.807 / 0.765	0.847 / 0.824	0.887 / 0.882	0.160 / 0.152	0.196 / 0.212	0.235 / 0.265
	2	12.5 / 29.0	18.0 / 29.0	27.0 / 30.0	0.327 / 0.235	0.423 / 0.353	0.500 / 0.471	0.070 / 0.059	0.100 / 0.094	0.127 / 0.156
	3	19.0 / 27.0	36.5 / 29.0	61.2 / 30.0	0.779 / 0.706	0.826 / 0.812	0.872 / 0.882	0.118 / 0.118	0.157 / 0.171	0.200 / 0.235
BUPA	1	30.8 / 29.0	43.2 / 29.0	59.0 / 30.0	0.441 / 0.357	0.516 / 0.429	0.594 / 0.571	0.095 / 0.100	0.134 / 0.158	0.196 / 0.250
	2	30.0 / 29.0	43.5 / 30.0	60.5 / 30.0	0.804 / 0.750	0.866 / 0.842	0.905 / 0.900	0.406 / 0.429	0.484 / 0.571	0.559 / 0.643
CENS	1	20.0 / 22.0	33.0 / 24.5	66.0 / 27.0	0.975 / 0.975	0.986 / 0.986	0.994 / 0.994	0.724 / 0.731	0.827 / 0.825	0.890 / 0.897
	2	19.0 / 27.0	24.0 / 28.5	36.0 / 30.0	0.110 / 0.103	0.173 / 0.175	0.276 / 0.269	0.006 / 0.006	0.014 / 0.014	0.025 / 0.025
CONT	1	24.5 / 24.0	46.0 / 26.0	67.0 / 27.0	0.743 / 0.726	0.807 / 0.790	0.859 / 0.869	0.523 / 0.512	0.603 / 0.610	0.675 / 0.698
	2	18.0 / 26.0	24.0 / 28.0	32.0 / 29.0	0.213 / 0.161	0.285 / 0.258	0.376 / 0.355	0.081 / 0.080	0.114 / 0.116	0.153 / 0.164
	3	18.2 / 27.0	25.5 / 28.0	39.0 / 29.0	0.139 / 0.120	0.204 / 0.184	0.278 / 0.265	0.079 / 0.076	0.124 / 0.130	0.175 / 0.185
IMAG	1	11.0 / 22.0	16.5 / 26.0	24.0 / 28.0	0.833 / 0.774	0.933 / 0.842	1.000 / 0.943	0.028 / 0.049	0.044 / 0.058	0.067 / 0.079
	2	9.0 / 24.0	14.0 / 29.0	31.0 / 30.0	1.000 / 0.987	1.000 / 0.997	1.000 / 1.000	0.022 / 0.029	0.028 / 0.042	0.050 / 0.063
	3	10.0 / 23.0	12.8 / 26.0	17.0 / 28.0	0.767 / 0.696	0.850 / 0.756	0.900 / 0.813	0.028 / 0.047	0.039 / 0.058	0.056 / 0.076
	4	10.0 / 26.0	15.0 / 28.0	17.0 / 29.0	0.300 / 0.090	0.367 / 0.150	0.467 / 0.243	0.000 / 0.007	0.006 / 0.011	0.011 / 0.021
	5	8.0 / 25.0	9.0 / 27.0	15.0 / 29.0	0.433 / 0.302	0.533 / 0.362	0.600 / 0.413	0.006 / 0.017	0.011 / 0.025	0.017 / 0.034
	6	13.0 / 27.0	16.2 / 28.0	20.0 / 29.0	0.900 / 0.873	0.933 / 0.916	0.967 / 0.949	0.067 / 0.098	0.083 / 0.116	0.106 / 0.152
	7	20.5 / 2.0	25.8 / 2.0	31.0 / 3.0	1.000 / 0.980	1.000 / 0.987	1.000 / 0.987	0.000 / 0.001	0.000 / 0.002	0.000 / 0.003
IRIS	1	21.0 / 2.0	28.0 / 3.0	44.0 / 12.0	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	21.5 / 30.0	28.5 / 30.0	40.0 / 30.0	0.933 / 0.800	0.956 / 1.000	0.956 / 1.000	0.000 / 0.000	0.011 / 0.000	0.023 / 0.050
	3	24.5 / 29.0	32.0 / 30.0	48.0 / 30.0	0.955 / 1.000	0.977 / 1.000	1.000 / 1.000	0.022 / 0.000	0.023 / 0.000	0.034 / 0.100
KD99	1	24.0 / 26.0	36.0 / 28.0	46.0 / 29.0	0.958 / 0.972	0.981 / 0.985	0.992 / 0.992	0.047 / 0.192	0.059 / 0.216	0.067 / 0.234
	2	21.0 / 22.0	31.2 / 24.0	43.0 / 27.0	0.200 / 0.013	0.476 / 0.105	0.539 / 0.117	0.000 / 0.000	0.001 / 0.001	0.002 / 0.003
	3	15.0 / 20.0	22.8 / 23.0	32.0 / 26.0	0.949 / 0.859	0.953 / 0.864	0.964 / 0.874	0.007 / 0.018	0.012 / 0.028	0.020 / 0.037
	4	24.0 / 18.0	35.0 / 22.0	48.0 / 25.0	0.188 / 0.042	0.324 / 0.203	0.440 / 0.325	0.000 / 0.000	0.000 / 0.001	0.002 / 0.002
	5	21.0 / 22.0	30.8 / 25.0	38.5 / 27.0	0.038 / 0.000	0.404 / 0.200	0.692 / 0.471	0.001 / 0.001	0.004 / 0.005	0.009 / 0.014
PIMA	1	25.0 / 29.0	38.0 / 29.0	53.8 / 30.0	0.812 / 0.780	0.869 / 0.860	0.911 / 0.920	0.444 / 0.481	0.540 / 0.593	0.646 / 0.704
	2	23.8 / 28.0	37.5 / 29.0	53.0 / 30.0	0.354 / 0.296	0.460 / 0.407	0.556 / 0.519	0.089 / 0.080	0.131 / 0.140	0.188 / 0.220
SHUT	1	17.0 / 25.0	23.0 / 27.0	36.0 / 29.0	0.838 / 0.837	0.941 / 0.942	0.976 / 0.974	0.009 / 0.007	0.022 / 0.023	0.067 / 0.066
	2	16.0 / 23.0	24.5 / 27.0	32.0 / 28.0	0.622 / 0.308	0.730 / 0.577	0.919 / 0.846	0.000 / 0.000	0.001 / 0.001	0.005 / 0.005
	3	17.0 / 24.0	24.0 / 25.5	40.0 / 27.0	0.568 / 0.538	0.670 / 0.654	0.803 / 0.795	0.002 / 0.002	0.008 / 0.008	0.037 / 0.037
	4	16.0 / 24.0	17.8 / 26.0	20.5 / 28.0	0.879 / 0.878	0.949 / 0.949	0.983 / 0.985	0.001 / 0.001	0.006 / 0.006	0.012 / 0.013
	5	29.5 / 2.0	41.5 / 24.5	59.0 / 28.0	0.991 / 0.986	0.994 / 0.993	0.998 / 0.996	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
	6	16.5 / 2.0	23.0 / 19.0	36.0 / 26.0	0.833 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
	7	20.0 / 2.0	26.5 / 5.0	33.5 / 14.0	0.818 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.057 / 0.058
THYD	1	22.0 / 28.0	33.2 / 29.0	49.0 / 30.0	0.753 / 0.767	0.823 / 0.788	0.871 / 0.863	0.004 / 0.006	0.008 / 0.010	0.022 / 0.023
	2	31.0 / 28.0	48.5 / 29.0	61.0 / 30.0	0.634 / 0.593	0.746 / 0.746	0.806 / 0.819	0.023 / 0.031	0.042 / 0.056	0.061 / 0.078
	3	32.0 / 28.0	46.0 / 30.0	61.5 / 30.0	0.916 / 0.891	0.949 / 0.934	0.978 / 0.967	0.046 / 0.048	0.129 / 0.140	0.208 / 0.248
WINE	1	13.0 / 30.0	17.5 / 30.0	27.8 / 30.0	0.943 / 0.833	0.963 / 1.000	0.981 / 1.000	0.037 / 0.000	0.056 / 0.083	0.084 / 0.167
	2	24.2 / 29.0	41.0 / 30.0	66.0 / 30.0	0.828 / 0.714	0.875 / 0.857	0.922 / 1.000	0.010 / 0.000	0.021 / 0.000	0.031 / 0.091
	3	14.0 / 29.0	18.0 / 30.0	31.8 / 30.0	0.977 / 1.000	1.000 / 1.000	1.000 / 1.000	0.009 / 0.000	0.017 / 0.000	0.026 / 0.071
WISC	1	36.0 / 24.0	49.0 / 27.0	67.0 / 29.0	0.952 / 0.932	0.959 / 0.955	0.967 / 0.977	0.023 / 0.000	0.033 / 0.042	0.047 / 0.083
	2	21.0 / 27.0	26.0 / 28.0	34.0 / 29.0	0.953 / 0.917	0.967 / 0.958	0.977 / 1.000	0.033 / 0.023	0.041 / 0.045	0.048 / 0.068

Table G.7: Classwise Quartile Results - PGEC

-		Soln Size / Participants			Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3	Q1	MED	Q3
BOST	1	16.0 / 6.0	25.0 / 8.0	41.0 / 10.0	0.000 / 0.000	0.183 / 0.176	0.761 / 0.750	0.000 / 0.000	0.067 / 0.061	0.461 / 0.471
	2	17.0 / 6.0	27.0 / 9.5	45.0 / 14.5	0.035 / 0.000	0.502 / 0.529	0.968 / 0.944	0.033 / 0.029	0.449 / 0.441	0.937 / 0.940
	3	16.0 / 6.0	23.2 / 8.0	41.0 / 10.0	0.000 / 0.000	0.087 / 0.062	0.542 / 0.588	0.000 / 0.000	0.026 / 0.029	0.303 / 0.290
BUPA	1	16.0 / 5.0	26.0 / 8.0	48.0 / 15.0	0.008 / 0.000	0.773 / 0.786	1.000 / 1.000	0.011 / 0.000	0.749 / 0.750	1.000 / 1.000
	2	17.0 / 4.0	26.5 / 8.0	48.0 / 16.0	0.000 / 0.000	0.251 / 0.250	0.989 / 1.000	0.000 / 0.000	0.227 / 0.214	0.992 / 1.000
CENS	1	14.0 / 11.0	19.8 / 12.0	54.5 / 16.0	0.078 / 0.079	0.706 / 0.707	0.993 / 0.993	0.028 / 0.027	0.406 / 0.411	0.966 / 0.964
	2	11.5 / 11.0	19.5 / 13.0	39.0 / 16.0	0.034 / 0.036	0.594 / 0.589	0.972 / 0.973	0.007 / 0.007	0.294 / 0.293	0.922 / 0.921
CONT	1	15.8 / 8.0	21.2 / 11.0	44.8 / 14.0	0.013 / 0.016	0.180 / 0.180	0.622 / 0.642	0.007 / 0.000	0.111 / 0.111	0.532 / 0.574
	2	16.0 / 9.0	23.0 / 11.0	39.5 / 15.0	0.000 / 0.000	0.123 / 0.111	0.667 / 0.656	0.001 / 0.000	0.092 / 0.091	0.516 / 0.532
	3	16.0 / 9.0	24.0 / 12.0	45.0 / 21.0	0.008 / 0.000	0.316 / 0.327	0.800 / 0.818	0.011 / 0.011	0.290 / 0.281	0.775 / 0.783
IMAG	1	15.0 / 4.0	21.0 / 6.0	36.5 / 8.0	0.000 / 0.000	0.000 / 0.003	0.200 / 0.189	0.000 / 0.000	0.000 / 0.001	0.044 / 0.048
	2	14.0 / 3.0	21.2 / 4.0	54.0 / 5.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.010	0.000 / 0.000	0.000 / 0.000	0.000 / 0.002
	3	17.0 / 6.0	24.5 / 8.5	38.0 / 11.0	0.000 / 0.000	0.100 / 0.144	0.700 / 0.732	0.000 / 0.000	0.019 / 0.020	0.333 / 0.372
	4	13.5 / 8.0	23.2 / 9.0	39.0 / 12.0	0.033 / 0.023	0.183 / 0.183	0.667 / 0.707	0.017 / 0.012	0.167 / 0.148	0.617 / 0.604
	5	18.5 / 6.0	26.5 / 7.5	39.0 / 9.0	0.000 / 0.000	0.000 / 0.007	0.133 / 0.111	0.000 / 0.000	0.000 / 0.003	0.039 / 0.035
	6	19.0 / 7.0	29.0 / 9.0	46.0 / 11.0	0.000 / 0.000	0.083 / 0.094	0.533 / 0.527	0.000 / 0.000	0.056 / 0.043	0.300 / 0.335
	7	4.0 / 1.0	10.0 / 1.0	18.0 / 1.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
IRIS	1	13.0 / 1.0	26.0 / 1.0	53.0 / 1.0	0.000 / 0.000	0.023 / 0.000	0.581 / 0.600	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	20.0 / 3.5	34.0 / 5.0	53.0 / 7.0	0.878 / 0.800	1.000 / 1.000	1.000 / 1.000	0.149 / 0.125	0.525 / 0.500	0.902 / 0.900
	3	20.0 / 3.0	31.5 / 4.0	55.0 / 5.0	0.022 / 0.000	0.477 / 0.400	0.909 / 1.000	0.000 / 0.000	0.011 / 0.000	0.261 / 0.300
KD99	1	10.0 / 7.0	22.0 / 8.0	35.0 / 10.0	0.000 / 0.000	0.318 / 0.340	0.923 / 0.956	0.000 / 0.000	0.007 / 0.067	0.247 / 0.384
	2	10.5 / 7.0	23.2 / 9.0	39.0 / 10.0	0.000 / 0.000	0.200 / 0.122	0.736 / 0.456	0.000 / 0.000	0.012 / 0.008	0.330 / 0.289
	3	13.0 / 6.0	19.0 / 7.5	43.0 / 9.0	0.000 / 0.000	0.000 / 0.001	0.356 / 0.296	0.000 / 0.000	0.000 / 0.000	0.017 / 0.019
	4	10.0 / 7.0	17.2 / 8.0	32.5 / 10.0	0.000 / 0.000	0.017 / 0.029	0.593 / 0.512	0.000 / 0.000	0.005 / 0.001	0.294 / 0.227
	5	10.0 / 6.0	16.0 / 8.0	31.5 / 10.0	0.000 / 0.000	0.000 / 0.000	0.288 / 0.157	0.000 / 0.000	0.000 / 0.000	0.021 / 0.020
PIMA	1	16.0 / 10.5	26.2 / 15.0	52.0 / 25.0	0.189 / 0.180	0.781 / 0.780	0.998 / 1.000	0.178 / 0.148	0.749 / 0.774	0.996 / 1.000
	2	16.0 / 9.0	23.0 / 13.0	43.5 / 20.0	0.004 / 0.000	0.251 / 0.226	0.822 / 0.852	0.002 / 0.000	0.219 / 0.220	0.811 / 0.820
SHUT	1	16.0 / 9.0	23.2 / 12.0	54.5 / 13.0	0.075 / 0.073	0.441 / 0.443	0.764 / 0.765	0.013 / 0.014	0.174 / 0.172	0.547 / 0.560
	2	13.0 / 3.0	21.2 / 5.0	37.0 / 7.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	13.0 / 5.0	24.0 / 7.0	42.5 / 9.0	0.000 / 0.000	0.000 / 0.000	0.015 / 0.026	0.000 / 0.000	0.000 / 0.000	0.016 / 0.016
	4	16.0 / 9.0	25.2 / 12.0	44.0 / 14.0	0.023 / 0.025	0.485 / 0.498	0.933 / 0.928	0.018 / 0.019	0.269 / 0.272	0.624 / 0.623
	5	16.5 / 1.0	30.8 / 3.0	58.0 / 9.0	0.000 / 0.000	0.000 / 0.000	0.002 / 0.005	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
	6	11.0 / 1.0	24.0 / 1.0	49.0 / 4.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	16.5 / 1.0	30.5 / 1.0	51.0 / 8.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
THYD	1	13.0 / 5.0	20.0 / 7.0	30.5 / 9.0	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	10.0 / 9.0	15.0 / 11.0	24.5 / 13.0	0.000 / 0.000	0.777 / 0.819	1.000 / 1.000	0.000 / 0.000	0.547 / 0.561	0.991 / 0.990
	3	10.0 / 8.0	17.5 / 11.0	26.0 / 13.0	0.001 / 0.000	0.302 / 0.283	0.999 / 1.000	0.000 / 0.000	0.055 / 0.036	0.996 / 0.996
WINE	1	17.0 / 5.0	28.0 / 6.0	45.0 / 7.0	0.415 / 0.333	0.860 / 0.833	1.000 / 1.000	0.065 / 0.077	0.280 / 0.333	0.664 / 0.667
	2	17.0 / 4.0	27.0 / 6.0	48.0 / 7.0	0.188 / 0.143	0.609 / 0.571	0.922 / 0.875	0.010 / 0.000	0.229 / 0.273	0.552 / 0.600
	3	16.0 / 2.0	26.0 / 4.0	48.8 / 5.0	0.000 / 0.000	0.045 / 0.000	0.465 / 0.400	0.000 / 0.000	0.000 / 0.000	0.017 / 0.000
WISC	1	16.0 / 4.0	26.0 / 6.0	48.0 / 9.0	0.772 / 0.759	0.970 / 0.977	0.997 / 1.000	0.038 / 0.042	0.314 / 0.333	0.847 / 0.870
	2	16.0 / 4.0	28.0 / 6.0	50.5 / 8.0	0.153 / 0.130	0.686 / 0.667	0.962 / 0.958	0.003 / 0.000	0.030 / 0.023	0.228 / 0.241

G.4 Neural Network Classifiers

Table G.8: Classwise Quartile Results - LP

-		Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3
BOST	1	0.821 / 0.765	0.840 / 0.812	0.860 / 0.875	0.062 / 0.061	0.079 / 0.103	0.101 / 0.143
	2	0.697 / 0.588	0.742 / 0.667	0.775 / 0.765	0.144 / 0.147	0.160 / 0.188	0.173 / 0.235
	3	0.800 / 0.706	0.819 / 0.789	0.833 / 0.882	0.055 / 0.057	0.062 / 0.086	0.075 / 0.118
BUPA	1	0.555 / 0.429	0.578 / 0.500	0.594 / 0.643	0.145 / 0.100	0.162 / 0.158	0.168 / 0.250
	2	0.832 / 0.750	0.838 / 0.842	0.855 / 0.900	0.406 / 0.357	0.422 / 0.500	0.445 / 0.571
CENS	1	0.990 / 0.989	0.991 / 0.991	0.992 / 0.991	0.677 / 0.681	0.698 / 0.698	0.728 / 0.728
	2	0.272 / 0.272	0.302 / 0.302	0.323 / 0.319	0.008 / 0.009	0.009 / 0.009	0.010 / 0.011
CONT	1	0.599 / 0.590	0.659 / 0.645	0.719 / 0.726	0.296 / 0.293	0.375 / 0.383	0.480 / 0.500
	2	0.169 / 0.125	0.253 / 0.219	0.327 / 0.344	0.042 / 0.036	0.073 / 0.080	0.101 / 0.116
	3	0.401 / 0.367	0.483 / 0.465	0.575 / 0.566	0.238 / 0.234	0.304 / 0.304	0.379 / 0.403
IMAG	1	0.967 / 0.973	0.967 / 0.990	1.000 / 0.993	0.000 / 0.004	0.006 / 0.008	0.017 / 0.016
	2	0.067 / 0.030	0.767 / 0.768	1.000 / 1.000	0.000 / 0.000	0.000 / 0.001	0.000 / 0.004
	3	0.867 / 0.789	0.900 / 0.821	0.900 / 0.839	0.017 / 0.036	0.028 / 0.044	0.039 / 0.050
	4	0.767 / 0.690	0.800 / 0.738	0.800 / 0.757	0.000 / 0.002	0.047 / 0.058	0.161 / 0.166
	5	0.700 / 0.634	0.833 / 0.728	0.900 / 0.755	0.028 / 0.034	0.033 / 0.044	0.044 / 0.052
	6	1.000 / 0.997	1.000 / 1.000	1.000 / 1.000	0.006 / 0.017	0.011 / 0.023	0.033 / 0.033
	7	1.000 / 0.990	1.000 / 0.990	1.000 / 0.990	0.000 / 0.000	0.000 / 0.000	0.000 / 0.002
IRIS	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.956 / 1.000	0.956 / 1.000	1.000 / 1.000	0.023 / 0.000	0.034 / 0.100	0.287 / 0.300
	3	0.511 / 0.600	0.932 / 0.800	0.955 / 1.000	0.000 / 0.000	0.022 / 0.000	0.023 / 0.000
KD99	1	0.992 / 0.986	0.997 / 0.988	0.998 / 0.990	0.039 / 0.191	0.048 / 0.200	0.057 / 0.217
	2	0.006 / 0.003	0.054 / 0.128	0.305 / 0.159	0.000 / 0.000	0.001 / 0.002	0.005 / 0.007
	3	0.798 / 0.635	0.971 / 0.889	0.980 / 0.916	0.001 / 0.007	0.002 / 0.012	0.008 / 0.016
	4	0.106 / 0.068	0.464 / 0.408	0.760 / 0.503	0.000 / 0.002	0.001 / 0.003	0.006 / 0.007
	5	0.000 / 0.000	0.038 / 0.093	0.423 / 0.257	0.000 / 0.000	0.001 / 0.002	0.003 / 0.005
PIMA	1	0.864 / 0.800	0.878 / 0.860	0.889 / 0.900	0.369 / 0.333	0.398 / 0.407	0.411 / 0.481
	2	0.589 / 0.519	0.602 / 0.593	0.631 / 0.667	0.111 / 0.100	0.122 / 0.140	0.136 / 0.200
SHUT	1	0.977 / 0.976	0.980 / 0.979	0.987 / 0.986	0.034 / 0.034	0.063 / 0.062	0.255 / 0.255
	2	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	4	0.526 / 0.534	0.923 / 0.925	0.964 / 0.964	0.056 / 0.055	0.087 / 0.085	0.090 / 0.090
	5	0.000 / 0.000	0.000 / 0.000	0.002 / 0.001	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	6	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	0.000 / 0.000	0.000 / 0.000	0.273 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
THYD	1	0.860 / 0.849	0.903 / 0.890	0.925 / 0.918	0.008 / 0.008	0.009 / 0.010	0.013 / 0.016
	2	0.021 / 0.017	0.086 / 0.136	0.445 / 0.435	0.001 / 0.002	0.002 / 0.003	0.003 / 0.005
	3	0.996 / 0.990	0.996 / 0.991	0.997 / 0.993	0.320 / 0.380	0.460 / 0.524	0.553 / 0.584
WINE	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	1.000 / 0.875	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
WISC	1	0.972 / 0.955	0.975 / 0.977	0.977 / 0.977	0.005 / 0.000	0.005 / 0.000	0.009 / 0.042
	2	0.991 / 0.958	0.995 / 1.000	0.995 / 1.000	0.023 / 0.023	0.025 / 0.023	0.028 / 0.045

Table G.9: Classwise Quartile Results - MLP

-		Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3
BOST	1	0.857 / 0.765	0.900 / 0.824	0.933 / 0.938	0.036 / 0.059	0.053 / 0.091	0.115 / 0.149
	2	0.653 / 0.500	0.819 / 0.706	0.859 / 0.765	0.087 / 0.094	0.117 / 0.176	0.153 / 0.242
	3	0.805 / 0.688	0.839 / 0.765	0.880 / 0.882	0.039 / 0.030	0.052 / 0.088	0.088 / 0.152
BUPA	1	0.000 / 0.000	0.602 / 0.500	0.669 / 0.643	0.000 / 0.000	0.112 / 0.150	0.156 / 0.250
	2	0.844 / 0.750	0.888 / 0.850	1.000 / 1.000	0.331 / 0.357	0.398 / 0.500	1.000 / 1.000
CENS	1	0.990 / 0.990	1.000 / 1.000	1.000 / 1.000	0.650 / 0.649	1.000 / 1.000	1.000 / 1.000
	2	0.000 / 0.000	0.000 / 0.000	0.350 / 0.351	0.000 / 0.000	0.000 / 0.000	0.010 / 0.010
CONT	1	0.602 / 0.557	0.667 / 0.656	0.789 / 0.787	0.221 / 0.237	0.320 / 0.338	0.550 / 0.540
	2	0.000 / 0.000	0.000 / 0.000	0.390 / 0.375	0.000 / 0.000	0.000 / 0.000	0.126 / 0.134
	3	0.248 / 0.220	0.588 / 0.560	0.727 / 0.707	0.135 / 0.135	0.321 / 0.339	0.455 / 0.467
IMAG	1	0.000 / 0.000	0.433 / 0.498	0.967 / 0.970	0.000 / 0.000	0.003 / 0.007	0.139 / 0.128
	2	0.133 / 0.180	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.005	0.022 / 0.030
	3	0.000 / 0.003	0.867 / 0.828	0.933 / 0.883	0.000 / 0.005	0.033 / 0.044	0.094 / 0.109
	4	0.000 / 0.000	0.517 / 0.535	0.833 / 0.803	0.000 / 0.000	0.006 / 0.016	0.067 / 0.092
	5	0.000 / 0.000	0.233 / 0.243	0.833 / 0.728	0.000 / 0.000	0.019 / 0.032	0.183 / 0.174
	6	0.000 / 0.000	0.033 / 0.048	1.000 / 0.986	0.000 / 0.000	0.000 / 0.003	0.033 / 0.042
	7	0.067 / 0.050	1.000 / 0.990	1.000 / 0.993	0.000 / 0.000	0.011 / 0.011	0.056 / 0.069
IRIS	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.911 / 0.800	0.978 / 1.000	0.978 / 1.000	0.000 / 0.000	0.000 / 0.000	0.040 / 0.125
	3	0.955 / 0.800	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.011 / 0.000	0.034 / 0.100
KD99	1	0.998 / 0.995	0.999 / 0.995	0.999 / 0.997	0.014 / 0.170	0.026 / 0.188	0.039 / 0.199
	2	0.000 / 0.000	0.145 / 0.002	0.858 / 0.063	0.000 / 0.000	0.000 / 0.000	0.002 / 0.001
	3	0.982 / 0.920	0.982 / 0.924	0.999 / 0.935	0.000 / 0.004	0.001 / 0.006	0.006 / 0.013
	4	0.000 / 0.000	0.829 / 0.520	0.938 / 0.647	0.000 / 0.000	0.000 / 0.004	0.001 / 0.005
	5	0.000 / 0.000	0.000 / 0.000	0.135 / 0.171	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
PIMA	1	0.864 / 0.810	0.889 / 0.880	0.999 / 1.000	0.324 / 0.370	0.369 / 0.481	0.975 / 1.000
	2	0.025 / 0.000	0.631 / 0.519	0.676 / 0.630	0.001 / 0.000	0.111 / 0.120	0.136 / 0.190
SHUT	1	0.998 / 0.998	1.000 / 1.000	1.000 / 1.000	0.011 / 0.010	0.084 / 0.083	0.925 / 0.925
	2	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	4	0.000 / 0.000	0.924 / 0.927	1.000 / 1.000	0.000 / 0.000	0.002 / 0.001	0.008 / 0.007
	5	0.000 / 0.000	0.000 / 0.000	0.998 / 0.998	0.000 / 0.000	0.000 / 0.000	0.001 / 0.001
	6	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
THYD	1	0.000 / 0.000	0.758 / 0.692	0.925 / 0.849	0.000 / 0.000	0.001 / 0.002	0.005 / 0.005
	2	0.000 / 0.000	0.000 / 0.003	0.880 / 0.842	0.000 / 0.000	0.000 / 0.000	0.004 / 0.012
	3	0.996 / 0.988	0.998 / 0.993	1.000 / 1.000	0.081 / 0.124	0.489 / 0.560	1.000 / 1.000
WINE	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	1.000 / 0.875	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
WISC	1	0.977 / 0.955	0.982 / 0.977	0.987 / 0.977	0.005 / 0.000	0.005 / 0.042	0.009 / 0.043
	2	0.991 / 0.957	0.995 / 0.958	0.995 / 1.000	0.013 / 0.023	0.018 / 0.023	0.023 / 0.045

G.5 Deterministic Classifiers

Table G.10: Classwise Quartile Results - 1R

-		Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3
BOST	1	0.808 / 0.706	0.814 / 0.813	0.820 / 0.882	0.069 / 0.061	0.087 / 0.116	0.095 / 0.143
	2	0.619 / 0.471	0.635 / 0.558	0.647 / 0.647	0.144 / 0.147	0.149 / 0.166	0.163 / 0.206
	3	0.840 / 0.706	0.843 / 0.824	0.846 / 0.882	0.108 / 0.114	0.121 / 0.119	0.125 / 0.152
BUPA	1	0.531 / 0.286	0.596 / 0.400	0.617 / 0.500	0.196 / 0.300	0.254 / 0.350	0.274 / 0.450
	2	0.726 / 0.550	0.746 / 0.650	0.804 / 0.700	0.383 / 0.500	0.404 / 0.600	0.469 / 0.714
CENS	1	0.997 / 0.997	0.997 / 0.997	0.997 / 0.997	0.835 / 0.841	0.835 / 0.841	0.835 / 0.841
	2	0.165 / 0.159	0.165 / 0.159	0.165 / 0.159	0.003 / 0.003	0.003 / 0.003	0.003 / 0.003
CONT	1	0.681 / 0.597	0.690 / 0.672	0.702 / 0.738	0.450 / 0.415	0.462 / 0.457	0.468 / 0.524
	2	0.004 / 0.000	0.056 / 0.031	0.063 / 0.094	0.001 / 0.009	0.026 / 0.032	0.026 / 0.036
	3	0.498 / 0.469	0.502 / 0.500	0.512 / 0.620	0.357 / 0.333	0.359 / 0.366	0.365 / 0.402
IMAG	1	0.900 / 0.815	0.900 / 0.815	0.900 / 0.815	0.150 / 0.155	0.150 / 0.155	0.150 / 0.155
	2	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.028 / 0.025	0.028 / 0.025	0.028 / 0.025
	3	0.467 / 0.385	0.467 / 0.385	0.467 / 0.385	0.033 / 0.044	0.033 / 0.044	0.033 / 0.044
	4	0.533 / 0.453	0.533 / 0.453	0.533 / 0.453	0.078 / 0.101	0.078 / 0.101	0.078 / 0.101
	5	0.433 / 0.272	0.433 / 0.272	0.433 / 0.272	0.022 / 0.042	0.022 / 0.042	0.022 / 0.042
	6	0.733 / 0.637	0.733 / 0.637	0.733 / 0.637	0.044 / 0.057	0.044 / 0.057	0.044 / 0.057
	7	0.600 / 0.513	0.600 / 0.513	0.600 / 0.513	0.033 / 0.063	0.033 / 0.063	0.033 / 0.063
IRIS	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.889 / 0.800	0.922 / 1.000	0.956 / 1.000	0.011 / 0.000	0.034 / 0.000	0.034 / 0.100
	3	0.932 / 0.800	0.933 / 1.000	0.977 / 1.000	0.023 / 0.000	0.040 / 0.000	0.057 / 0.100
KD99	1	0.989 / 0.992	0.989 / 0.992	0.989 / 0.992	0.087 / 0.254	0.087 / 0.254	0.087 / 0.254
	2	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	0.948 / 0.859	0.948 / 0.859	0.948 / 0.859	0.004 / 0.003	0.004 / 0.003	0.004 / 0.003
	4	0.396 / 0.572	0.396 / 0.572	0.396 / 0.572	0.005 / 0.005	0.005 / 0.005	0.005 / 0.005
	5	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
PIMA	1	0.887 / 0.820	0.908 / 0.880	0.922 / 0.940	0.469 / 0.481	0.514 / 0.593	0.537 / 0.593
	2	0.463 / 0.407	0.485 / 0.407	0.531 / 0.519	0.078 / 0.060	0.092 / 0.120	0.113 / 0.180
SHUT	1	0.939 / 0.939	0.939 / 0.939	0.939 / 0.939	0.020 / 0.020	0.020 / 0.020	0.020 / 0.020
	2	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	0.068 / 0.128	0.068 / 0.128	0.068 / 0.128	0.000 / 0.001	0.000 / 0.001	0.000 / 0.001
	4	0.992 / 0.991	0.992 / 0.991	0.992 / 0.991	0.057 / 0.057	0.057 / 0.057	0.057 / 0.057
	5	0.997 / 0.990	0.997 / 0.990	0.997 / 0.990	0.001 / 0.001	0.001 / 0.001	0.001 / 0.001
	6	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
THYD	1	0.796 / 0.781	0.796 / 0.781	0.796 / 0.781	0.009 / 0.012	0.009 / 0.012	0.009 / 0.012
	2	0.775 / 0.712	0.775 / 0.712	0.775 / 0.712	0.019 / 0.022	0.019 / 0.022	0.019 / 0.022
	3	0.980 / 0.973	0.980 / 0.973	0.980 / 0.973	0.109 / 0.156	0.109 / 0.156	0.109 / 0.156
WINE	1	0.962 / 0.833	0.962 / 1.000	0.963 / 1.000	0.131 / 0.091	0.140 / 0.167	0.148 / 0.182
	2	0.656 / 0.571	0.696 / 0.714	0.750 / 0.714	0.052 / 0.091	0.083 / 0.091	0.104 / 0.182
	3	0.791 / 0.800	0.907 / 0.800	0.930 / 1.000	0.009 / 0.000	0.051 / 0.000	0.052 / 0.077
WISC	1	0.972 / 0.955	0.975 / 0.977	0.975 / 0.977	0.146 / 0.167	0.150 / 0.174	0.151 / 0.208
	2	0.849 / 0.792	0.850 / 0.826	0.854 / 0.833	0.025 / 0.023	0.025 / 0.023	0.028 / 0.045

Table G.11: Classwise Quartile Results - NB

-		Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3
BOST	1	0.747 / 0.647	0.753 / 0.697	0.760 / 0.882	0.124 / 0.086	0.125 / 0.131	0.127 / 0.182
	2	0.487 / 0.444	0.520 / 0.500	0.529 / 0.588	0.150 / 0.094	0.152 / 0.179	0.157 / 0.219
	3	0.826 / 0.750	0.843 / 0.849	0.852 / 0.938	0.161 / 0.118	0.174 / 0.194	0.192 / 0.229
WISC	1	0.954 / 0.932	0.957 / 0.955	0.959 / 0.977	0.019 / 0.000	0.024 / 0.042	0.024 / 0.042
	2	0.976 / 0.958	0.976 / 0.958	0.981 / 1.000	0.041 / 0.023	0.043 / 0.045	0.046 / 0.068
CENS	1	0.878 / 0.880	0.878 / 0.880	0.878 / 0.880	0.374 / 0.367	0.374 / 0.367	0.374 / 0.367
	2	0.626 / 0.633	0.626 / 0.633	0.626 / 0.633	0.122 / 0.120	0.122 / 0.120	0.122 / 0.120
CONT	1	0.467 / 0.443	0.478 / 0.459	0.490 / 0.516	0.208 / 0.150	0.213 / 0.235	0.215 / 0.256
	2	0.525 / 0.452	0.540 / 0.531	0.565 / 0.581	0.245 / 0.209	0.248 / 0.279	0.258 / 0.300
	3	0.474 / 0.440	0.488 / 0.455	0.497 / 0.500	0.281 / 0.247	0.289 / 0.298	0.299 / 0.337
IMAG	1	0.967 / 0.892	0.967 / 0.892	0.967 / 0.892	0.028 / 0.015	0.028 / 0.015	0.028 / 0.015
	2	1.000 / 0.990	1.000 / 0.990	1.000 / 0.990	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	0.300 / 0.214	0.300 / 0.214	0.300 / 0.214	0.017 / 0.021	0.017 / 0.021	0.017 / 0.021
	4	0.867 / 0.870	0.867 / 0.870	0.867 / 0.870	0.028 / 0.057	0.028 / 0.057	0.028 / 0.057
	5	0.767 / 0.789	0.767 / 0.789	0.767 / 0.789	0.122 / 0.135	0.122 / 0.135	0.122 / 0.135
	6	0.933 / 0.866	0.933 / 0.866	0.933 / 0.866	0.000 / 0.003	0.000 / 0.003	0.000 / 0.003
	7	1.000 / 0.987	1.000 / 0.987	1.000 / 0.987	0.000 / 0.001	0.000 / 0.001	0.000 / 0.001
IRIS	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.956 / 0.800	0.956 / 1.000	0.956 / 1.000	0.034 / 0.000	0.034 / 0.000	0.046 / 0.100
	3	0.909 / 0.800	0.932 / 1.000	0.932 / 1.000	0.023 / 0.000	0.023 / 0.000	0.023 / 0.100
KD99	1	0.864 / 0.944	0.864 / 0.944	0.864 / 0.944	0.027 / 0.119	0.027 / 0.119	0.027 / 0.119
	2	0.381 / 0.028	0.381 / 0.028	0.381 / 0.028	0.003 / 0.003	0.003 / 0.003	0.003 / 0.003
	3	0.961 / 0.876	0.961 / 0.876	0.961 / 0.876	0.005 / 0.011	0.005 / 0.011	0.005 / 0.011
	4	0.802 / 0.852	0.802 / 0.852	0.802 / 0.852	0.051 / 0.030	0.051 / 0.030	0.051 / 0.030
	5	0.942 / 0.714	0.942 / 0.714	0.942 / 0.714	0.037 / 0.032	0.037 / 0.032	0.037 / 0.032
BUPA	1	0.756 / 0.643	0.774 / 0.714	0.813 / 0.786	0.564 / 0.500	0.595 / 0.565	0.631 / 0.600
	2	0.369 / 0.400	0.405 / 0.435	0.436 / 0.500	0.188 / 0.214	0.227 / 0.286	0.244 / 0.357
PIMA	1	0.842 / 0.820	0.847 / 0.840	0.853 / 0.880	0.378 / 0.333	0.387 / 0.396	0.398 / 0.481
	2	0.602 / 0.519	0.613 / 0.604	0.622 / 0.667	0.147 / 0.120	0.153 / 0.160	0.158 / 0.180
SHUT	1	0.974 / 0.975	0.974 / 0.975	0.974 / 0.975	0.265 / 0.263	0.265 / 0.263	0.265 / 0.263
	2	0.676 / 0.308	0.676 / 0.308	0.676 / 0.308	0.013 / 0.012	0.013 / 0.012	0.013 / 0.012
	3	0.561 / 0.590	0.561 / 0.590	0.561 / 0.590	0.008 / 0.009	0.008 / 0.009	0.008 / 0.009
	4	0.626 / 0.625	0.626 / 0.625	0.626 / 0.625	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	5	0.996 / 0.995	0.996 / 0.995	0.996 / 0.995	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	6	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.002 / 0.002	0.002 / 0.002	0.002 / 0.002
THYD	1	0.828 / 0.822	0.828 / 0.822	0.828 / 0.822	0.003 / 0.006	0.003 / 0.006	0.003 / 0.006
	2	0.372 / 0.316	0.372 / 0.316	0.372 / 0.316	0.008 / 0.010	0.008 / 0.010	0.008 / 0.010
	3	0.992 / 0.987	0.992 / 0.987	0.992 / 0.987	0.440 / 0.496	0.440 / 0.496	0.440 / 0.496
WINE	1	0.981 / 0.833	0.981 / 1.000	0.981 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.984 / 0.875	0.984 / 1.000	0.984 / 1.000	0.010 / 0.000	0.010 / 0.000	0.010 / 0.091
	3	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.009 / 0.000	0.009 / 0.000	0.009 / 0.000

Table G.12: Classwise Quartile Results - J48

-		Detection Rate (Train / Test)			False Positive Rate (Train / Test)		
DB	Class	Q1	MED	Q3	Q1	MED	Q3
BOST	1	0.933 / 0.706	0.944 / 0.812	0.960 / 0.882	0.023 / 0.086	0.023 / 0.103	0.030 / 0.182
	2	0.859 / 0.412	0.875 / 0.629	0.897 / 0.706	0.037 / 0.118	0.047 / 0.188	0.057 / 0.206
	3	0.947 / 0.813	0.953 / 0.875	0.960 / 0.882	0.029 / 0.086	0.041 / 0.118	0.055 / 0.121
BUPA	1	0.641 / 0.429	0.738 / 0.500	0.803 / 0.643	0.034 / 0.150	0.045 / 0.175	0.056 / 0.300
	2	0.944 / 0.700	0.955 / 0.825	0.966 / 0.850	0.197 / 0.357	0.262 / 0.500	0.359 / 0.571
CENS	1	0.995 / 0.984	0.995 / 0.984	0.995 / 0.984	0.388 / 0.538	0.388 / 0.538	0.388 / 0.538
	2	0.612 / 0.462	0.612 / 0.462	0.612 / 0.462	0.005 / 0.016	0.005 / 0.016	0.005 / 0.016
CONT	1	0.774 / 0.590	0.782 / 0.618	0.797 / 0.639	0.149 / 0.280	0.158 / 0.296	0.162 / 0.317
	2	0.609 / 0.313	0.649 / 0.344	0.663 / 0.355	0.093 / 0.164	0.098 / 0.188	0.102 / 0.209
	3	0.706 / 0.429	0.718 / 0.455	0.729 / 0.469	0.146 / 0.272	0.156 / 0.287	0.173 / 0.304
IMAG	1	0.967 / 0.933	0.967 / 0.933	0.967 / 0.933	0.011 / 0.008	0.011 / 0.008	0.011 / 0.008
	2	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.001	0.000 / 0.001	0.000 / 0.001
	3	0.967 / 0.873	0.967 / 0.873	0.967 / 0.873	0.011 / 0.034	0.011 / 0.034	0.011 / 0.034
	4	0.967 / 0.877	0.967 / 0.877	0.967 / 0.877	0.000 / 0.017	0.000 / 0.017	0.000 / 0.017
	5	0.933 / 0.779	0.933 / 0.779	0.933 / 0.779	0.011 / 0.041	0.011 / 0.041	0.011 / 0.041
	6	1.000 / 0.976	1.000 / 0.976	1.000 / 0.976	0.000 / 0.002	0.000 / 0.002	0.000 / 0.002
	7	0.967 / 0.933	0.967 / 0.933	0.967 / 0.933	0.000 / 0.002	0.000 / 0.002	0.000 / 0.002
IRIS	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.956 / 0.800	0.978 / 1.000	0.978 / 1.000	0.011 / 0.000	0.017 / 0.000	0.023 / 0.100
	3	0.955 / 0.800	0.966 / 1.000	0.977 / 1.000	0.011 / 0.000	0.011 / 0.000	0.022 / 0.100
KD99	1	1.000 / 0.983	1.000 / 0.983	1.000 / 0.983	0.001 / 0.118	0.001 / 0.118	0.001 / 0.118
	2	0.978 / 0.271	0.978 / 0.271	0.978 / 0.271	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	1.000 / 0.947	1.000 / 0.947	1.000 / 0.947	0.000 / 0.008	0.000 / 0.008	0.000 / 0.008
	4	0.994 / 0.808	0.994 / 0.808	0.994 / 0.808	0.000 / 0.013	0.000 / 0.013	0.000 / 0.013
	5	0.750 / 0.043	0.750 / 0.043	0.750 / 0.043	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
PIMA	1	0.853 / 0.780	0.921 / 0.820	0.929 / 0.920	0.220 / 0.346	0.240 / 0.388	0.295 / 0.444
	2	0.705 / 0.556	0.760 / 0.611	0.780 / 0.654	0.071 / 0.080	0.079 / 0.180	0.147 / 0.220
SHUT	1	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	2	0.946 / 0.923	0.946 / 0.923	0.946 / 0.923	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	3	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	4	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	5	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	6	0.667 / 0.000	0.667 / 0.000	0.667 / 0.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
	7	1.000 / 1.000	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.000 / 0.000
THYD	1	1.000 / 0.945	1.000 / 0.945	1.000 / 0.945	0.000 / 0.003	0.000 / 0.003	0.000 / 0.003
	2	0.958 / 0.921	0.958 / 0.921	0.958 / 0.921	0.001 / 0.005	0.001 / 0.005	0.001 / 0.005
	3	0.999 / 0.991	0.999 / 0.991	0.999 / 0.991	0.028 / 0.072	0.028 / 0.072	0.028 / 0.072
WINE	1	0.981 / 1.000	0.981 / 1.000	1.000 / 1.000	0.000 / 0.000	0.000 / 0.000	0.009 / 0.000
	2	0.984 / 0.875	0.984 / 1.000	0.984 / 1.000	0.010 / 0.000	0.010 / 0.045	0.010 / 0.091
	3	1.000 / 0.800	1.000 / 1.000	1.000 / 1.000	0.000 / 0.000	0.009 / 0.000	0.009 / 0.000
WISC	1	0.972 / 0.932	0.982 / 0.955	0.987 / 0.977	0.014 / 0.000	0.014 / 0.042	0.019 / 0.083
	2	0.981 / 0.917	0.986 / 0.958	0.986 / 1.000	0.013 / 0.023	0.018 / 0.045	0.028 / 0.068

G.6 Overall Comparisons

Table G.13: Accuracy Quartile Comparisons (Train / Test)

DB	Quartile	1R	CM-GE1	CM-GE2	J48	LP	MLP	NB	PGEC	RSS-GE	STD-GE
BOST	Q1	0.755 / 0.706	0.670 / 0.627	0.670 / 0.615	0.902 / 0.712	0.780 / 0.720	0.637 / 0.615	0.689 / 0.653	0.339 / 0.333	0.600 / 0.558	0.573 / 0.549
	MED	0.759 / 0.731	0.688 / 0.667	0.692 / 0.654	0.921 / 0.733	0.799 / 0.758	0.832 / 0.731	0.696 / 0.697	0.355 / 0.353	0.634 / 0.627	0.626 / 0.612
	Q3	0.765 / 0.769	0.707 / 0.694	0.709 / 0.700	0.941 / 0.796	0.809 / 0.788	0.860 / 0.784	0.709 / 0.745	0.422 / 0.422	0.665 / 0.673	0.670 / 0.673
BUPA	Q1	0.678 / 0.500	0.637 / 0.588	0.691 / 0.606	0.834 / 0.618	0.712 / 0.629	0.585 / 0.588	0.550 / 0.500	0.417 / 0.412	0.579 / 0.529	0.607 / 0.588
	MED	0.686 / 0.522	0.671 / 0.636	0.712 / 0.647	0.858 / 0.682	0.732 / 0.676	0.741 / 0.647	0.560 / 0.550	0.462 / 0.456	0.620 / 0.588	0.642 / 0.618
	Q3	0.690 / 0.588	0.697 / 0.676	0.730 / 0.706	0.879 / 0.727	0.739 / 0.735	0.775 / 0.714	0.577 / 0.588	0.583 / 0.588	0.648 / 0.647	0.678 / 0.672
CENS	Q1	0.945 / 0.945	0.919 / 0.918	0.930 / 0.931	0.971 / 0.951	0.947 / 0.946	0.937 / 0.937	0.862 / 0.864	0.130 / 0.131	0.598 / 0.600	0.937 / 0.937
	MED	0.945 / 0.945	0.932 / 0.933	0.937 / 0.937	0.971 / 0.951	0.948 / 0.948	0.937 / 0.937	0.862 / 0.864	0.700 / 0.701	0.687 / 0.688	0.937 / 0.937
	Q3	0.945 / 0.945	0.937 / 0.937	0.938 / 0.938	0.971 / 0.951	0.948 / 0.948	0.949 / 0.949	0.862 / 0.864	0.931 / 0.932	0.749 / 0.750	0.937 / 0.937
CONT	Q1	0.481 / 0.444	0.468 / 0.438	0.468 / 0.444	0.721 / 0.479	0.498 / 0.470	0.433 / 0.433	0.494 / 0.465	0.329 / 0.312	0.423 / 0.399	0.430 / 0.413
	MED	0.483 / 0.474	0.481 / 0.465	0.482 / 0.468	0.730 / 0.496	0.512 / 0.493	0.508 / 0.486	0.495 / 0.483	0.357 / 0.361	0.443 / 0.431	0.441 / 0.433
	Q3	0.485 / 0.500	0.495 / 0.493	0.493 / 0.489	0.743 / 0.504	0.522 / 0.521	0.553 / 0.528	0.498 / 0.507	0.418 / 0.412	0.464 / 0.463	0.460 / 0.465
IMAG	Q1	0.667 / 0.582	0.771 / 0.707	0.781 / 0.692	0.971 / 0.910	0.781 / 0.753	0.405 / 0.384	0.833 / 0.801	0.143 / 0.145	0.657 / 0.609	0.633 / 0.619
	MED	0.667 / 0.582	0.805 / 0.723	0.802 / 0.715	0.971 / 0.910	0.852 / 0.825	0.552 / 0.543	0.833 / 0.801	0.183 / 0.181	0.712 / 0.676	0.714 / 0.688
	Q3	0.667 / 0.582	0.829 / 0.744	0.819 / 0.731	0.971 / 0.910	0.933 / 0.900	0.686 / 0.673	0.833 / 0.801	0.233 / 0.226	0.752 / 0.715	0.752 / 0.724
IRIS	Q1	0.955 / 0.929	0.962 / 0.933	0.970 / 0.929	0.977 / 0.933	0.799 / 0.751	0.659 / 0.667	0.955 / 0.923	0.371 / 0.357	0.924 / 0.867	0.939 / 0.867
	MED	0.955 / 0.933	0.970 / 0.933	0.977 / 0.933	0.978 / 0.933	0.955 / 0.931	0.970 / 0.923	0.959 / 0.967	0.508 / 0.500	0.955 / 0.933	0.962 / 0.933
	Q3	0.962 / 1.000	0.977 / 1.000	0.978 / 1.000	0.985 / 1.000	0.970 / 0.933	0.992 / 0.933	0.962 / 1.000	0.667 / 0.667	0.963 / 1.000	0.970 / 1.000
KD99	Q1	0.958 / 0.897	0.930 / 0.881	0.946 / 0.877	1.000 / 0.937	0.905 / 0.831	0.976 / 0.903	0.896 / 0.883	0.015 / 0.040	0.896 / 0.852	0.889 / 0.802
	MED	0.958 / 0.897	0.956 / 0.888	0.957 / 0.883	1.000 / 0.937	0.968 / 0.894	0.983 / 0.918	0.896 / 0.883	0.393 / 0.360	0.924 / 0.876	0.909 / 0.865
	Q3	0.958 / 0.897	0.962 / 0.893	0.962 / 0.892	1.000 / 0.937	0.977 / 0.909	0.990 / 0.923	0.896 / 0.883	0.600 / 0.620	0.950 / 0.884	0.926 / 0.882
PIMA	Q1	0.757 / 0.684	0.683 / 0.662	0.682 / 0.649	0.822 / 0.724	0.773 / 0.727	0.651 / 0.658	0.761 / 0.724	0.403 / 0.390	0.565 / 0.542	0.651 / 0.649
	MED	0.761 / 0.714	0.722 / 0.701	0.719 / 0.688	0.844 / 0.752	0.783 / 0.766	0.789 / 0.724	0.766 / 0.747	0.599 / 0.597	0.621 / 0.597	0.654 / 0.649
	Q3	0.766 / 0.753	0.744 / 0.740	0.744 / 0.737	0.855 / 0.792	0.789 / 0.805	0.805 / 0.779	0.771 / 0.805	0.650 / 0.649	0.670 / 0.662	0.666 / 0.671
SHUT	Q1	0.947 / 0.946	0.785 / 0.784	0.863 / 0.864	1.000 / 1.000	0.871 / 0.874	0.784 / 0.792	0.920 / 0.922	0.185 / 0.177	0.736 / 0.737	0.787 / 0.794
	MED	0.947 / 0.946	0.882 / 0.882	0.936 / 0.936	1.000 / 1.000	0.903 / 0.904	0.908 / 0.908	0.920 / 0.922	0.456 / 0.452	0.791 / 0.787	0.804 / 0.811
	Q3	0.947 / 0.946	0.937 / 0.930	0.966 / 0.966	1.000 / 1.000	0.916 / 0.916	0.984 / 0.984	0.920 / 0.922	0.660 / 0.666	0.831 / 0.836	0.822 / 0.829
THYD	Q1	0.965 / 0.956	0.909 / 0.882	0.907 / 0.888	0.997 / 0.987	0.944 / 0.940	0.923 / 0.927	0.956 / 0.949	0.052 / 0.052	0.405 / 0.417	0.923 / 0.927
	MED	0.965 / 0.956	0.931 / 0.915	0.940 / 0.923	0.997 / 0.987	0.947 / 0.945	0.944 / 0.940	0.956 / 0.949	0.315 / 0.295	0.761 / 0.732	0.923 / 0.927
	Q3	0.965 / 0.956	0.952 / 0.942	0.954 / 0.942	0.997 / 0.987	0.966 / 0.960	0.984 / 0.973	0.956 / 0.949	0.923 / 0.927	0.863 / 0.839	0.923 / 0.927
WINE	Q1	0.831 / 0.750	0.925 / 0.889	0.919 / 0.842	0.988 / 0.895	1.000 / 0.944	1.000 / 0.944	0.981 / 0.944	0.410 / 0.389	0.813 / 0.778	0.843 / 0.784
	MED	0.834 / 0.778	0.943 / 0.944	0.932 / 0.895	0.988 / 0.944	1.000 / 1.000	1.000 / 1.000	0.988 / 0.972	0.506 / 0.500	0.868 / 0.833	0.887 / 0.882
	Q3	0.840 / 0.833	0.956 / 0.944	0.950 / 0.944	0.988 / 1.000	1.000 / 1.000	1.000 / 1.000	0.988 / 1.000	0.600 / 0.611	0.906 / 0.895	0.919 / 0.944
WISC	Q1	0.931 / 0.897	0.946 / 0.926	0.956 / 0.940	0.977 / 0.926	0.979 / 0.956	0.980 / 0.941	0.962 / 0.941	0.651 / 0.647	0.911 / 0.897	0.924 / 0.912
	MED	0.931 / 0.919	0.952 / 0.955	0.960 / 0.956	0.983 / 0.963	0.981 / 0.971	0.985 / 0.970	0.963 / 0.970	0.729 / 0.721	0.946 / 0.940	0.949 / 0.941
	Q3	0.933 / 0.926	0.959 / 0.970	0.965 / 0.971	0.984 / 0.970	0.984 / 0.985	0.989 / 0.971	0.965 / 0.971	0.852 / 0.838	0.961 / 0.956	0.964 / 0.970

Table G.14: Score Quartile Comparisons (Train / Test)

DB	Quartile	1R	CM-GE1	CM-GE2	J48	LP	MLP	NB	PGEC	RSS-GE	STD-GE
BOST	Q1	0.757 / 0.706	0.674 / 0.627	0.675 / 0.624	0.902 / 0.718	0.782 / 0.722	0.644 / 0.627	0.692 / 0.654	0.333 / 0.333	0.601 / 0.560	0.576 / 0.549
	MED	0.760 / 0.734	0.691 / 0.667	0.696 / 0.661	0.922 / 0.735	0.800 / 0.760	0.833 / 0.734	0.698 / 0.701	0.352 / 0.353	0.637 / 0.627	0.629 / 0.615
	Q3	0.767 / 0.772	0.710 / 0.701	0.712 / 0.703	0.941 / 0.798	0.810 / 0.791	0.860 / 0.784	0.711 / 0.745	0.419 / 0.418	0.667 / 0.678	0.672 / 0.675
BUPA	Q1	0.667 / 0.465	0.576 / 0.533	0.659 / 0.571	0.806 / 0.568	0.690 / 0.614	0.500 / 0.500	0.585 / 0.532	0.500 / 0.500	0.586 / 0.523	0.546 / 0.507
	MED	0.667 / 0.512	0.627 / 0.589	0.684 / 0.625	0.841 / 0.672	0.709 / 0.650	0.718 / 0.625	0.597 / 0.565	0.500 / 0.500	0.624 / 0.586	0.609 / 0.568
	Q3	0.674 / 0.550	0.664 / 0.642	0.705 / 0.671	0.869 / 0.700	0.717 / 0.698	0.757 / 0.696	0.606 / 0.607	0.503 / 0.500	0.650 / 0.642	0.646 / 0.636
CENS	Q1	0.581 / 0.578	0.534 / 0.534	0.549 / 0.548	0.803 / 0.723	0.632 / 0.631	0.500 / 0.500	0.752 / 0.756	0.500 / 0.500	0.720 / 0.723	0.500 / 0.500
	MED	0.581 / 0.578	0.582 / 0.585	0.579 / 0.581	0.803 / 0.723	0.647 / 0.646	0.500 / 0.500	0.752 / 0.756	0.513 / 0.513	0.736 / 0.735	0.500 / 0.500
	Q3	0.581 / 0.578	0.638 / 0.637	0.623 / 0.621	0.803 / 0.723	0.656 / 0.654	0.670 / 0.671	0.752 / 0.756	0.612 / 0.605	0.753 / 0.754	0.500 / 0.500
CONT	Q1	0.414 / 0.391	0.423 / 0.394	0.413 / 0.387	0.707 / 0.452	0.445 / 0.421	0.377 / 0.360	0.500 / 0.464	0.333 / 0.332	0.429 / 0.409	0.339 / 0.333
	MED	0.416 / 0.408	0.446 / 0.425	0.433 / 0.417	0.711 / 0.470	0.470 / 0.453	0.449 / 0.433	0.502 / 0.493	0.344 / 0.344	0.452 / 0.439	0.386 / 0.369
	Q3	0.419 / 0.423	0.464 / 0.454	0.453 / 0.446	0.729 / 0.479	0.488 / 0.478	0.486 / 0.487	0.505 / 0.513	0.374 / 0.374	0.468 / 0.466	0.416 / 0.416
IMAG	Q1	0.667 / 0.582	0.771 / 0.707	0.781 / 0.692	0.972 / 0.910	0.781 / 0.754	0.405 / 0.382	0.833 / 0.801	0.143 / 0.144	0.657 / 0.609	0.633 / 0.617
	MED	0.667 / 0.582	0.805 / 0.724	0.802 / 0.716	0.972 / 0.910	0.852 / 0.826	0.552 / 0.542	0.833 / 0.801	0.183 / 0.182	0.712 / 0.676	0.714 / 0.687
	Q3	0.667 / 0.582	0.829 / 0.744	0.819 / 0.732	0.972 / 0.910	0.933 / 0.900	0.686 / 0.674	0.833 / 0.801	0.233 / 0.227	0.752 / 0.715	0.752 / 0.724
IRIS	Q1	0.955 / 0.933	0.963 / 0.933	0.970 / 0.933	0.978 / 0.933	0.800 / 0.742	0.667 / 0.667	0.955 / 0.917	0.365 / 0.333	0.924 / 0.867	0.940 / 0.867
	MED	0.956 / 0.933	0.970 / 0.933	0.977 / 0.933	0.978 / 0.933	0.955 / 0.933	0.970 / 0.917	0.959 / 0.967	0.501 / 0.492	0.955 / 0.933	0.962 / 0.933
	Q3	0.963 / 1.000	0.978 / 1.000	0.978 / 1.000	0.985 / 1.000	0.970 / 0.933	0.993 / 0.933	0.963 / 1.000	0.665 / 0.667	0.963 / 1.000	0.970 / 1.000
KD99	Q1	0.467 / 0.485	0.585 / 0.467	0.576 / 0.428	0.944 / 0.610	0.434 / 0.396	0.495 / 0.395	0.790 / 0.683	0.207 / 0.204	0.475 / 0.396	0.369 / 0.324
	MED	0.467 / 0.485	0.639 / 0.505	0.609 / 0.483	0.944 / 0.610	0.509 / 0.478	0.570 / 0.496	0.790 / 0.683	0.270 / 0.252	0.528 / 0.444	0.378 / 0.370
	Q3	0.467 / 0.485	0.721 / 0.558	0.670 / 0.527	0.944 / 0.610	0.577 / 0.510	0.752 / 0.547	0.790 / 0.683	0.338 / 0.320	0.574 / 0.488	0.386 / 0.379
PIMA	Q1	0.690 / 0.623	0.579 / 0.548	0.614 / 0.574	0.807 / 0.689	0.733 / 0.678	0.500 / 0.500	0.726 / 0.698	0.498 / 0.500	0.554 / 0.529	0.501 / 0.500
	MED	0.701 / 0.652	0.644 / 0.617	0.656 / 0.627	0.818 / 0.709	0.741 / 0.726	0.753 / 0.678	0.729 / 0.707	0.500 / 0.500	0.603 / 0.588	0.510 / 0.500
	Q3	0.705 / 0.683	0.682 / 0.682	0.691 / 0.671	0.845 / 0.756	0.748 / 0.773	0.771 / 0.735	0.738 / 0.775	0.507 / 0.512	0.652 / 0.649	0.544 / 0.543
SHUT	Q1	0.428 / 0.435	0.789 / 0.750	0.786 / 0.782	0.945 / 0.846	0.268 / 0.265	0.143 / 0.143	0.833 / 0.785	0.151 / 0.146	0.402 / 0.382	0.163 / 0.167
	MED	0.428 / 0.435	0.863 / 0.820	0.849 / 0.826	0.945 / 0.846	0.278 / 0.277	0.285 / 0.285	0.833 / 0.785	0.170 / 0.164	0.508 / 0.517	0.217 / 0.217
	Q3	0.428 / 0.435	0.888 / 0.877	0.891 / 0.879	0.945 / 0.846	0.330 / 0.279	0.422 / 0.422	0.833 / 0.785	0.196 / 0.193	0.628 / 0.628	0.254 / 0.255
THYD	Q1	0.850 / 0.822	0.802 / 0.771	0.781 / 0.772	0.986 / 0.952	0.633 / 0.637	0.333 / 0.333	0.731 / 0.708	0.333 / 0.333	0.581 / 0.568	0.333 / 0.333
	MED	0.850 / 0.822	0.844 / 0.838	0.831 / 0.812	0.986 / 0.952	0.661 / 0.669	0.623 / 0.617	0.731 / 0.708	0.336 / 0.335	0.636 / 0.628	0.333 / 0.333
	Q3	0.850 / 0.822	0.874 / 0.870	0.868 / 0.852	0.986 / 0.952	0.789 / 0.779	0.890 / 0.848	0.731 / 0.708	0.372 / 0.382	0.710 / 0.667	0.333 / 0.333
WINE	Q1	0.840 / 0.738	0.933 / 0.897	0.927 / 0.859	0.988 / 0.905	1.000 / 0.952	1.000 / 0.944	0.983 / 0.944	0.381 / 0.349	0.828 / 0.794	0.851 / 0.810
	MED	0.847 / 0.806	0.947 / 0.944	0.942 / 0.917	0.988 / 0.933	1.000 / 1.000	1.000 / 1.000	0.988 / 0.976	0.480 / 0.467	0.877 / 0.857	0.897 / 0.897
	Q3	0.855 / 0.838	0.959 / 0.952	0.953 / 0.952	0.990 / 1.000	1.000 / 1.000	1.000 / 1.000	0.988 / 1.000	0.581 / 0.592	0.912 / 0.917	0.925 / 0.952
WISC	Q1	0.912 / 0.883	0.938 / 0.922	0.956 / 0.936	0.980 / 0.926	0.982 / 0.966	0.984 / 0.945	0.965 / 0.945	0.516 / 0.515	0.912 / 0.904	0.924 / 0.902
	MED	0.913 / 0.893	0.950 / 0.945	0.961 / 0.956	0.982 / 0.958	0.984 / 0.977	0.988 / 0.967	0.966 / 0.968	0.664 / 0.661	0.945 / 0.942	0.948 / 0.943
	Q3	0.914 / 0.905	0.958 / 0.967	0.967 / 0.972	0.983 / 0.978	0.985 / 0.979	0.990 / 0.977	0.968 / 0.976	0.839 / 0.831	0.963 / 0.966	0.964 / 0.966

Appendix H

KDD 99: Preprocessing of Attack Type Labels

Nominal label map with final group description and class number in brackets. Details of attack definitions can be found at MIT Lincon Laboratory web site.[1]

```
[0] => normal.  ( NORM => 0 )
[1] => snmpgetattack.  ( R2L => 1 )
[2] => named.  ( R2L => 1 )
[3] => xlock.  ( R2L => 1 )
[4] => smurf.  ( DOS => 2 )
[5] => ipsweep.  ( PROBE => 3 )
[6] => multihop.  ( R2L => 1 )
[7] => xsnoop.  ( R2L => 1 )
[8] => sendmail.  ( R2L => 1 )
[9] => guess_passwd.  ( R2L => 1 )
[10] => saint.  ( PROBE => 3 )
[11] => buffer_overflow.  ( U2R => 4 )
[12] => portsweep.  ( PROBE => 3 )
[13] => pod.  ( DOS => 2 )
[14] => apache2.  ( DOS => 2 )
[15] => phf.  ( R2L => 1 )
[16] => udpstorm.  ( DOS => 2 )
[17] => warezmaster.  ( R2L => 1 )
[18] => perl.  ( U2R => 4 )
[19] => satan.  ( PROBE => 3 )
[20] => xterm.  ( U2R => 4 )
[21] => mscan.  ( PROBE => 3 )
```

[1] http://www.ll.mit.edu/IST/ideval/data/data_index.html

```
[22] => processtable.  ( DOS => 2 )

[23] => ps.  ( U2R => 4 )
[24] => nmap.  ( PROBE => 3 )
[25] => rootkit.  ( U2R => 4 )
[26] => neptune.  ( DOS => 3 )
[27] => loadmodule.  ( U2R => 4 )
[28] => imap.  ( R2L => 1 )
[29] => back.  ( DOS => 3 )
[30] => httptunnel.  ( R2L => 1 )
[31] => worm.  ( R2L => 1 )
[32] => mailbomb.  ( DOS => 2 )
[33] => ftp_write.  ( R2L => 1 )
[34] => teardrop.  ( DOS => 3 )
[35] => land.  ( DOS => 3 )
[36] => sqlattack.  ( U2R => 4 )
[37] => snmpguess.  ( R2L => 1 )
[38] => warezclient.  ( R2L => 1 )
[39] => spy.  ( R2L => 1 )
```