Evaluation of Cluster Combination Functions for Mixture of Experts

R. Redhead

March 16, 2005

Contents

List of Tables **vi**

List of Figures **vii**

Abstract **ix**

Acknowledgements **x**

1 Introduction **1**

2 Background and Related Work **3**

2.1 Self-Organized Maps 4

2.2 Potential function clustering 5

2.3 MoEII 6

2.4 Information Gain and Mutually distant points 8

2.5 Maximum Volume Ellipsoid 9

3 Implementation of MoEIII **12**

3.1 K-Furthest Neighbors (KFN) 12

3.2 K-Nearest Neighbors (KNN) 12

3.3 Maximum Area Triangle (MAT) 13

3.4 Maximum polytope (MaxP) 14

3.5 Normalized Triangle (NT) 16

4 Experimental Procedure **18**

4.1 Parameters 18

4.2 Classification Data sets 19

4.3 Regression problems 20

4.4 Experimental Design 21

5 Results **22**
5.1 Classification . . . 23
5.1.1 BREAST . . . 23
5.1.1.1 Initial Tests . . . 23
5.1.1.2 Number of Clusters . . . 26
5.1.1.3 Higher Neighborhood sizes . . . 29
5.1.2 C-HEART . . . 32
5.1.2.1 Initial Tests . . . 32
5.1.2.2 Number of Clusters . . . 34
5.1.2.3 Higher Neighborhood sizes . . . 37
5.1.3 IONO . . . 40
5.1.3.1 Initial Tests . . . 40
5.1.3.2 Number of Clusters . . . 43
5.1.3.3 Higher Neighborhood sizes . . . 46
5.1.4 LIVER . . . 49
5.1.4.1 Initial Tests . . . 49
5.1.4.2 Number of Clusters . . . 52
5.1.4.3 Higher Neighborhood sizes . . . 54
5.2 Regression . . . 56
5.2.1 Lorenz . . . 56
5.2.1.1 Initial Tests . . . 56
5.2.1.2 Number of Clusters . . . 57
5.2.1.3 Higher Neighborhood sizes . . . 58
5.2.2 Model IV . . . 58
5.2.2.1 Initial Tests . . . 58
5.2.2.2 Number of Clusters . . . 59
5.2.2.3 Higher Neighborhood sizes . . . 60

6 Conclusion **62**

Bibliography **65**

A A chapter in the appendix **67**

B The next appendix **68**

C Glossary **69**

List of Tables

4.1 Benchmark Data Sets 19
4.2 Regression Data Sets 20

5.1 BREAST Set Previous Results 23
5.2 Error Rates for BREAST set 26
5.3 C-HEART Set Previous Results 32
5.4 Error Rates for C-HEART set 34
5.5 IONO Set Previous Results 40
5.6 Mean Error Rates for IONO set 43
5.7 LIVER Set Previous Results 49
5.8 Error Rates for LIVER set 52

6.1 Initial Test Winners 62

List of Figures

2.1 Information Gain 9

5.1 Box-plot of Test Minimums for BREAST data set 24
5.2 Box-plot of False Positives for BREAST data set 24
5.3 Box-plot of False Negatives for BREAST data set 25
5.4 Box-plot of Total Error Rates for BREAST data set 26
5.5 Plot of Test mins 27
5.6 Plot of False Positive Rates 27
5.7 Plot of False Negative Rates 28
5.8 Plot of Total Error Rates. 29
5.9 Plot of Test mins 30
5.10 Plot of False Positive Rates 30
5.11 Plot of False Negative Rates 31
5.12 Plot of Total Error Rates. 31
5.13 Box-plot of Test Minimums for C-HEART data set 32
5.14 Box-plot of False Positives for C-HEART data set 33
5.15 Box-plot of False Negatives for C-HEART data set 33
5.16 Box-plot of Total Error for C-HEART data set 34
5.17 Plot of Test mins. 35
5.18 Plot of False Positive Rates. 35
5.19 Plot of a) Test mins b) False Positive Rates c) False Negative Rates and d) Total Error Rates. 36
5.20 Plot of Total Error Rates. 37
5.21 Plot of Test mins. 38
5.22 Plot of False Positive Rates. 38
5.23 Plot of False Negative Rates. 39
5.24 Plot of Total Error Rates. 39
5.25 Box-plot of Test Minimums for IONO data set 41
5.26 Box-plot of False Positives for IONO data set 41

5.27 Box-plot of False Negatives for IONO data set 42
5.28 Box-plot of Total Error rates for IONO data set 42
5.29 Plot of Test mins. 44
5.30 Plot of False Positive Rates. 44
5.31 Plot of False Negative Rates. 45
5.32 Plot of Total Error Rates. 46
5.33 Plot of Test mins. 47
5.34 Plot of False Positive Rates. 47
5.35 Plot of False Negative Rates. 48
5.36 Plot of Total Error Rates. 49
5.37 Box-plot of Test Minimums for LIVER data set 50
5.38 Box-plot of False Positives for LIVER data set 50
5.39 Box-plot of False Negatives for LIVER data set 51
5.40 Box-plot of Total Errors for LIVER data set 51
5.41 Plot of Test mins. 52
5.42 Plot of False Positive Rates. 53
5.43 Plot of False Negative Rates. 53
5.44 Plot of Total Error Rates. 54
5.45 Plot of Test mins. 55
5.46 Plot of False Positive Rates. 55
5.47 Plot of False Negative Rates. 56
5.48 Plot of Total Error Rates. 56
5.49 Box-plot of Test Minimums for Lorenz data set 57
5.50 Plot of Test Minimums for Lorenz data set 58
5.51 Plot of Test Minimums for Lorenz data set 59
5.52 Box-plot of Test Minimums for Model IV data set 59
5.53 Plot of Test Minimums for Model IV data set 60
5.54 Plot of Test Minimums for Model IV data set 61

Abstract

The Mixtures of Experts (MoE) model provides the basis for building modular neural network solutions. In this work we are interested in methods for decomposing the input before forwarding to the MoE architecture. By doing so we are able to define the number of experts from the data itself. Specific schemes are shown to be appropriate for regression and classification problems, where each appear to have different preferences.This is your abstract.

Acknowledgements

The authors acknowledge support from NSERC and CFI New Opportunities Grants from the Canadian Government.

Chapter 1

Introduction

Jordan and Jacobs [2] proposed a Mixture of Experts (MoE) architecture in which, if you are given enough *a priori* knowledge, you can design a number of expert networks to divide up a problem. A gate network then mixes or switches between experts depending on the global context of the input. In a more general case, where there is little *a priori* knowledge, how do we choose inputs that will encourage the formation of complementary experts? We must also determine what inputs the gating network receives, where the gate requires a global view of the problem.

The MoEII architecture, proposed by Tang *et al.* [1], determines experts by preprocessing and partitioning the data. To do so a SOM preprocessing step was utilized. Specifically, the SOM was trained and partitioned with each MoE expert associated with a SOM partition. Thus, the number of experts and the respective inputs would be uniquely defined. In the case of the gate, the SOM neuron denoting the centroid of each partition defines the gate input. The basic motivation being that the gate should receive a global 'summary' of the problem whereas the individual experts receive a detailed view of specific partitions.

Although providing a process for deriving the required number of experts independent of any *a priori* knowledge and successfully avoiding over learning on a set of benchmark classification problems, there are drawbacks to the MoEII approach. In particular the SOM partitions used as input to each expert tend to be self similar - a natural consequence of the SOM topological ordering property. In this work we instead use a clustering preprocessing step in which the basic objective is to identify the optimal combination of cluster centroids on which MoE experts are built. such a scheme is synonymous with describing SOM partitions in terms of a set of clusters and asking what combinations of clusters, irrespective of partition, provides the best basis for building the MoE architecture or MoEIII. In the following, five schemes for combining clusters and therefore building the MoE architecture are investigated (Chapter 3.1). All provide a significant improvement over MoEII for a range of benchmark problems (Section III). We begin by introducing the MoEII and MoEII architectures, Sections II.A and II.B respectively.

Chapter 2

Background and Related Work

Traditionally preprocessing involves performing operations on the input data, such as normalization or isolating functional dependencies. Ramamurti and Ghosh [6] preprocess the data by projecting it into higher dimensional space using radial basis kernels, this is known to increase the likelihood that the problem will be linearly separable. The motivation behind this was to reduce the height of Hierarchical Mixture of Experts (HME) and thus reduce the computational cost in training. While this did produce a speed up, the preprocessing step did not explicitly define the height. This means that the structure of the network is defined by the programmer and not the nature of the problem, this is supervised and not a part of the architecture itself.

Wong *et al.*[12] utilizes functional dependencies to reduce the number of inputs that the neural network receives. They found this to be accurate in addition to providing a speed up to the network. Reducing the number of inputs reduces the size and complexity of the over all network. Although this is a preprocessing step that aids the design of the network, there are problems. First functional dependency is only applicable to symbolic attributes, so continuous attributes must be quantised. This may not be an impediment in may instances but it results in a loss of resolution and requires some arbitrary choice in boundaries. Second is that there could be examples in the training set that operate on different function dependencies than those in the test set. Therefore the training set must be carefully chosen such that important input attributes are not lost.

An unsupervised alternative would be to use clustering on the data. Clustering can give important hints about how we should choose our parameters or other critical features such as the number of experts for MoEs. However, clustering is usually performed as a step that is separate from the learning architecture. Kuncheva [4] offers an architecture that is similar to a committee of committees where the number of clusters in the data set decides the number of classifiers. The clustering function divides the input space up into a number of regions, and in each region the top performing classifier is nominated to label the inputs in that region. This presents a hierarchical divide and conquer technique that is reminiscent of HMEs, even if it isn't neural network based. Integrating this idea into a MoE architecture would produce a semi-unsupervised architecture such as that proposed by Tang *et al.* [1].

2.1 Self-Organized Maps

Kohonen's Self-Organized Map (SOM)[9] is a competitive network which characterizes input patterns through lateral interactions. A SOM is a sheet-like network where neighborhoods consist of spatially similar nodes. By adapting these nodes to frequent input patterns it produces an abstraction of the input space in accordance to how the data is distributed. Because neighboring neurons learn neighboring sections of the input space, SOMs not only learn the distribution but also the topology of the data.

The learning algorithm for the SOM is an unsupervised procedure where nodes weights are initialized to random values. When a pattern x is presented to the network the winning neuron n_c is identified by the minimum Euclidean distance of all nodes to the input pattern, $||x - n_c|| = \min_i ||x - n_i||$. The weights of the winner and its neighborhood are adjusted as $n_i(t+1) = n_i(t) + h_{ci}(t)\{x(t) - n_i(t)\}$, where t is the epoch and $h_{ci}(t)$ is the neighborhood function. Here the neighborhood is defined by the link distance function.

The link distance D between two points vectors n_i and n_j from a set of S vectors is:

- $D_{ij} = 0$, if $i = j$
- $D_{ij} = 1$, if $\sqrt{\Sigma(P_i - P_j)^2}$ is ≤ 1
- $D_{ij} = 2$, if k exists, and $D_{ik} = D_{kj} = 1$
- $D_{ij} = 3$, if $k_1$and k_2 exist, and $D_{ik_1} = D_{k_1k_2} = D_{k_2j} = 1$
- $D_{ij} = N$, if $k_1 ... k_N$ exist, and $D_{ik_1} = D_{k_1k_2} = ... = D_{k_Nj} = 1$
- $D_{ij} = S$, if none of the above conditions apply.

The selection and re-weighting steps repeat until some stopping criteria is reached. The stopping criteria usually involves convergence, such as the absolute squared weight changes being smaller than some small value ϵ, or it can involve stopping once a large number of epochs are reached. The former criteria offers the best characterization of the data.

A SOM had been utilized in the MoEII (Section 2.3) and MoEIII (Chapter 3) architectures due to the fact that the SOM has no fuzzy regions making it easily grouped by a second level clustering function. If there are superfluous nodes, as a result of outliers or mislabeled examples, they are quickly pruned. This means that the SOM is robust and insensitive to noise.

2.2 Potential function clustering

The purpose of clustering is to determine the natural groupings of data from a large data set. Ideally this will produce an accurate representation of the system's behavior. There are many clustering techniques such as C-means, K-means, etc... in this paper we use a modified version of the Mountain Method [14], called the Potential function, developed by Chiu [5].

In both methods various points are assigned a potential value based on their proximity to data points. When such a point is chosen as a cluster center its potential is reduced to zero, and the surrounding points have their potential reduced based on their proximity to it. The point with the next highest potential is selected and the process repeats until the stopping conditions are met.

The difference in Chiu's algorithm is that instead of using an arbitrary grid system of points, the data points themselves are assigned potential values based on their proximity to other data points. In the mountain method a grid resolution of R and data dimension d, we would have R^d grid points for which to make calculations. By using the data points themselves the number of points is completely independent of the dimensions of the data. Also we avoid the difficulties of choosing a resolution, a factor that is the source of a trade off between accuracy and calculation time.

The potential method can be used as the basis for a fuzzy model calculation, making it useful as a robust function. Also because it does not involve any iterative nonlinear optimization and because computation time is linear with respect to the dimensions of the data it is a fast algorithm to run.

The potential method does indeed yield fast performance and produces good characterization of the data. While the methods ability to determine the number of clusters by itself is useful and robust, the number of clusters was determined arbitrarily and the neighborhood radius manipulated to give the desired results. This was done so that it would be easier to cross compare later results across different data sets. The effects of varying the number of clusters were determined for each data model to observe if this had any interference.

In the MoEII architecture the potential function was used as a second level clustering function to cluster the nodes of a Self Organized Map (SOM). In the SOM the first N dimensions represent the input space of the data, and the last M-N dimensions represent the output space. Clustering was performed on the full M dimensions even though the later learning methods only receive the input space as input. Clustering on the full dimensions of the problem diversifies and differentiates the cluster centers in ways that may not be visible without the output space. A simple example could be where the input spaces of a class 1 cluster and a class 0 cluster overlap. If we viewed this from the input space alone we would certainly end up with one cluster to represent them both instead of two. Where the SOM represents memory, clustering represents the abstraction and characterization of similar events. Clustering establishes that there are different types of events, the Neural Network is designed to be a predictive model. It establishes a connection between cause and effect, then predicts the outcome of future events.

2.3 MoEII

The implementation of MoEII involves a standard mixture of experts (MoE) architecture, where the input data is preprocessed using a Self Organized Map (SOM) to characterize the data. The SOM nodes are clustered using the potential function proposed by Chiu[5], this scheme does not require *a priori* specification of clusters, however, any clustering algorithm with this property would be appropriate. Once region centroids are chosen, nodes are assigned to clusters based on a nearest neighbor routine.

One expert is assigned to each SOM partition, therefore if there are K partitions then there are also K experts. During training, examples are presented to the SOM and the SOM outputs are computed. The inputs for expert i are taken from the corresponding SOM partition. The gate network receives an input from each partition, the partition centroid. Thus the gate network as K inputs and K outputs.

Each expert i is a two layer network consisting of an input layer (from the SOM) and a single perceptron output layer. A sigmoid transfer function is used for classification problems and a simple linear function for regression problems. Random weights were assigned to the weight vectors between SOM nodes and experts, for w_{ji}, $i = 1...K$, $j = 1...|c_i|$. For the Gate weight matrix, W_g, the elements are a_{ij} $i = 1...K$, $j = 1...K$.

Every expert i has a vector of weights $\bar{w}_i$ which is a randomly initialized $|c_i| \times 1$

column vector. The gate network has a $K \times K$ weight matrix W_g, also randomly initialized, where rows correspond to experts and columns correspond to clusters.

- $y_i = \bar{w_i'} * \bar{x}$

  $\bar{y} = \frac{1}{(1+\exp(-\bar{y}))}$

- $\bar{u} = W_g * clsum(\bar{x})$

  $\bar{g} = \frac{\exp(\bar{u})}{\sum_{i=1}^{K} \exp(u_i)}$

- $Y_{out} = \sum_{i=1}^{K} y_i * g_i$

  $e = d - Y_{out}$

- $h_i = \frac{g_i * y^d * (1-y_i)^{1-d}}{\sum_{j=1}^{K} g_j * y^d * (1-y_j)^{1-d}}$

- $\bar{dw} = \eta * h_i * ec * (d - y_i) * \bar{x}$

  $\bar{m} = \tau * (\bar{w}_i(t) - \bar{w}_i(t-1))$

  $\bar{w}_i(t+1) = \bar{w}_i(t) + \bar{dw} + \bar{m}$

- $dW_g = \eta * (h - g) * clsum(\bar{x})'$

  $M = \tau * (W_g(t) - W_g(t-1))$

  $W_g(t+1) = W_g(t) + dW_g + M$

η is the learning rate, τ is the momentum rate, and ec is the error correction for the smaller class in the case of class imbalance. For the larger class $ec = 1$, and for the smaller class it is equal to #large class examples / #small class examples ($ec \geq 1$). Where $\bar{cc}_i$ is the weight vector of the cluster center of cluster i, $\bar{n}_{ji}$ is the weight vector of the jth node in cluster i. The weight factors are computed as:

$$wf_{ji} = \| \bar{n}_{ji} - \bar{cc}_i \|^{-1} / \sum_{j=1}^{|c_i|} \| \bar{n}_{ji} - \bar{cc}_i \|^{-1}$$

And the column vector of cluster summaries is computed as:

$$clsum_i(\bar{x}) = \sum_{j=1}^{|c_i|} wf_{ji} * \| \bar{x} - \bar{n}_{ji} \|$$

Error rates were lower with the MoEII model, there was strong correlation between the learning and test curves, meaning there was no over fitting, and the model was not very sensitive to sub-optimal parameters. "For MoE-II, a deviation from the optimal parameter setting would only cause minor degradation in the over all performance"

[1]. Also the new architecture showed a much more modular solution; where as the standard MoE had complex solutions involving several experts for a mapping. Thus transparency was improved.

A problem with the MoEII architecture has to do with the nature of clustering. The mere fact that nodes belong to a single cluster means that they are in some way self-similar. Forwarding every SOM output of a cluster to a single expert is redundant, since it is unlikely that the individual nodes offer any significant information gain above that of the cluster centroid. As stated before the cluster centroid offers a summary of the entire cluster, so it should be sufficient to only use the centroids as inputs thus reducing the complexity of Experts.

2.4 Information Gain and Mutually distant points

Navigation by distant points of reference is a concept as old as travel itself. On the ocean, with no landmarks, sailors navigated by the stars using a sextant to measure the angle between the horizon and the sun or stars. The measured angle between where the stars rise and fall at the beginning of the journey and where they would later would give the navigator a reasonably accurate description of where they were. A larger change in the angle would offer them more information gain.

If we take each degree to have an equal probability of 1/360, then the angle between reference points is the cumulative probabilities of those degrees in between, therefore $P(\theta) = \frac{\theta}{360}$. The information gain of θ is defined by equation 2.1.

$$I(\theta) = -P^{\text{posterior}}(\theta) * \log_2 \frac{P^{\text{prior}}(\theta)}{P^{\text{posterior}}(\theta)} \tag{2.1}$$

Here $P^{\text{prior}} = \frac{1}{360}$ since it is a single degree on the compass. This means that equation 2.1 has an ever growing information gain as the angle increases, as can be seen in Fig 2.1.

Computing angles between abstract points can be difficult. However there is another way to ensure that the reference points are mutually distant from each other, and that is to maximize the enclosed area of the point of view and reference points. This forces the points further apart from each other thus widening the angle between them. A way to optimize the angles for every point (not just the point of view) would be to find a combination of points such that when the distances are normalized, the points form as close to a unit shape as possible (e.g. a unit triangle for 3 points). These methods do not require us to directly calculate the angles between the points, but rather utilize the distance norms instead. The advantage that we have over the

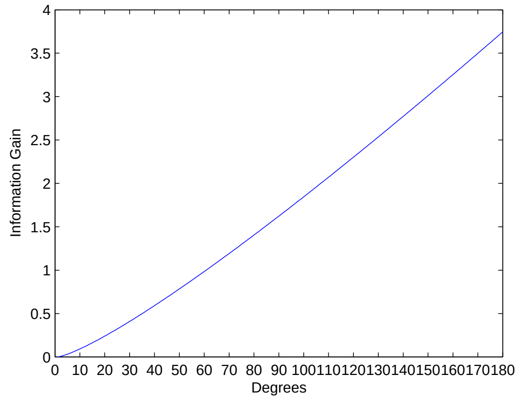


Figure 2.1: Information Gain

sailors of old is that we know the exact positions of these points from the input parameters of our datasets and calculating the Euclidean Norm is trivial.

2.5 Maximum Volume Ellipsoid

Later we will discuss an architecture that utilizes the maximum enclosed area to choose neighborhoods. For Neighborhood sizes of three this is easily done using Heron's Formula, because three points form a triangle and a triangle always exists on a 2-dimensional hyperplane regardless of the actual dimensions of the problem. To expand the method to neighborhood sizes greater than three a method was needed to calculate the area or volume of higher dimensional polytopes.

Calculating the exact volume of a polytope can be quite difficult and time consuming since the permutation of the edges affects the shape of the polytope. However the exact volume is not needed, only an approximate value so that different polytopes can be compared. Ellipsoids, on the other hand, have good geometric properties that make handling and performing operations on them less expensive[13]. For these reasons the Dual Reduced Newton (DRN) algorithm [8] for the Maximum Volume Ellipsoid (MVE) problem was chosen to approximate the volume of the polytopes.

The MVE problem is often used as a subroutine for other algorithms and therefore special attention has been paid to its optimization. The DRN algorithm is just such an optimization that has demonstrated high accuracy on very large examples, m=30,000 and n=30 [8]. Its performance is more than sufficient for our purposes where $m \leq 7$.

If A is the matrix that contains the coordinates of the cluster centers as its columns and its dimensions are n x m, m is the number of cluster centers or vertexes and n is the dimension of the problem space, then the rank of [A; $\bar{e}$] ($\bar{e}$ is a vector of ones) must be equal to or greater than n+1. What this means is that the number of points or vertexes must be greater than the dimensions and that A must have no 0 rows.

Given the above preconditions the Dual Reduced Newton Algorithm has the following form:

Algorithm 1 DRN-Direction (u, t, θ)

Given the point (u,t) where u, t > 0 and θ where $\theta \geq 0$

```
1. form and factorize the matrix $M^{-2}(u) = [2 * (AUA^T - \frac{Auu^T A^T}{e^T u})]$
2. form the matrix $\Sigma(u) = (A - \frac{Aue^T}{e^T u})^T M^2(u)(A - \frac{Aue^T}{e^T u})$
3. form $\bigtriangledown_u h(u) = -2(\frac{\Sigma(u)}{e^T u} + \Sigma(u) \circ \Sigma(u))$ and factorize $(\bigtriangledown_u h(u) - U^{-1}T)$
4. solve the equations:
    (a) $\triangle u = (\bigtriangledown_u h(u) - U^{-1}T^{-1})^{-1}(r_1 - U^{-1}r_2)$
    (b) $\triangle t = U^{-1}r_2 - U^{-1}T \triangle u$
```

Algorithm 2 DRN Algorithm

```
Step 0: Initialization. Set $r \leftarrow 0.99$. Choose initial values of $(u^0, t^0)$ satisfying
    $u^0, t^0 > 0$. Set $(u, t) \leftarrow (u^0, t^0)$.
Step 1: Check Stopping Criteria. $OBJ = -\ln \det[M(u)]$.
    • If $||e - h(u) - t|| \leq \epsilon_1$ and $\frac{u^T t}{OBJ} \leq \epsilon_2$ then STOP. Return $Q = [M(u)]^2$,
      $c = [M(u)]^{-1} z(u)$ and OBJ.
Step 2: Compute Direction. Set $\theta \leftarrow \frac{u^T t}{10m}$. Compute using DRN-Direction$(u, t, \theta)$.
Step 3: Step-size Computation and Step. Compute $\bar{\beta} \leftarrow \max\{\beta | (u, t) + \beta(\triangle u, \triangle t)\}$
    and $\tilde{\beta} \leftarrow \max\{r\bar{\beta}, 1\}$. Set $(u, t) \leftarrow (u, t) + \tilde{\beta}(\triangle u, \triangle t)$. Go to Step 1.
```

The rank$\begin{vmatrix} W' \\ \bar{e} \end{vmatrix} \geq n + 1$ precondition on the DRN algorithm that can cause complications however. Most problems have many dimensions and the advantage of the gated network is that very few inputs are needed to perform well. Therefore a constraint such as this is counter productive; we do not want to select more inputs/points/vertexes than the dimensions, especially since some problems can have dozens of dimensions. Therefore the dimensions of the problem have to be scaled down.

The reason for this constraint is quite simple, if you want to have a polytope in n dimensions then you have to have more than n points, for example in 2-dimensions 2 points cannot define a 2-dimensional shape but 3 points can. If however you have m points, where m $<$ n, then A should be able to define a shape in an m-1 dimensional hyperplane. The discovered hyperplane could be rotated and translated until it exists only in an m-1 dimensional space with an m-1 dimensional coordinate system. This can be accomplished using Multi-dimensional scaling (Mds)[11] a method most commonly used in statistics to display high dimensional data as two or three dimensional charts or graphs so that a human can understand them.

Chapter 3

Implementation of MoEIII

The differing factor of the MoEIII model is that only the cluster summaries are forwarded to experts. As before there are K experts for K clusters, however each expert receives inputs from several different cluster summaries. The experts are decoupled from the SOM nodes and are therefore free to receive combinations of clustered stimuli. For every expert k there are F empirically chosen inputs from F different cluster summaries the first of which is from the kth cluster, the other F-1 are chosen by one of five cluster functions. The various functions and their descriptions are as follows:

3.1 K-Furthest Neighbors (KFN)

With centroid k as the first choice, choose the F-1 furthest neighbors as the remaining inputs. Calculate the distance matrix D for centroids where $D_{ij} = \parallel c_i - c_j \parallel$

- For each expert $k \to K$
- $inputs_1 = c_k$ The first input is the kth centroid
- for $j = 2 \to F$

  1. $c_x = \max(D_k)$. Find c_x such that D_k is the largest distance value in row k of D.
  2. $inputs_j = c_x$. Assign the new input.
  3. $D_{kx} = 0$

3.2 K-Nearest Neighbors (KNN)

With centroid k as the first choice, choose the F-1 nearest neighbors as the remaining inputs. Calculate the distance matrix D for centroids where $D_{ij} = \parallel c_i - c_j \parallel$

- For each expert $k \rightarrow K$
- $inputs_1 = c_k$ The first input is the kth centroid
- for $j = 2 \rightarrow F$

    1. $c_x = \min(D_k)$. Find c_x such that D_k is the smallest distance value in row k of D.
    2. $inputs_j = c_x$. Assign the new input.
    3. $D_{kx} = \infty$

3.3 Maximum Area Triangle (MAT)

The Maximum Area Triangle method is based on the idea that points that are mutually distant from each other will provide more information gain when it comes time to classify examples associated with the primary centroid k. The primary centroid is the only mandatory centroid or vertex that the triangles must contain. The combination of centroids is chosen based on the further criteria that the area that they enclose must be the maximum possible enclosed area by the centroid k. It collects more and more triangles until the number of unique vertexes required is reached.

The method uses triangles because the enclose area is easily calculated using Herron's Formula. All that is needed is the Euclidean distance between each point. These distances form the sides of the triangle. In this single step we calculate the enclosed area without having to worry about the dimensions of the problem, because triangles are always two dimensional shapes regardless of the hyperplane in which they exist.

Select the combination of three centroids, which contains centroid k, with the largest enclosed area. To select more than three inputs, select a centroid such that when connected to two existing inputs forms the next largest area.

- Generate all combinations C. Where $|C| = \binom{K}{3}$
- Calculate areas A using Herron's formula...

  For each combination c_x in C

  $side_1 = \|c_{x1} - c_{x2}\|$

  $side_2 = \|c_{x2} - c_{x3}\|$

  $side_3 = \|c_{x3} - c_{x1}\|$

$s = \frac{side_1 + side_2 + side_3}{2}$

$A_x = \sqrt{s * (s - side_1) * (s - side_2) * (s - side_3)}$

- For each expert $k \rightarrow K$
- $C' = \{c_x | \forall x[x \in I \wedge k \in c_x]\}$ where I is the Set Index. C' is all combinations in C such that all combinations in C' contain k.
- $A'_x = \max(A')$ where A' are the areas in A that correspond to C'. Find x such that A'_x is max in A'.
- $inputs = C'_x$. inputs is the combination that corresponds to A'
- while $(|inputs| < F)$

  1. $C'' = \{c_x | \forall x[x \in I \wedge \exists j[j \in inputs \wedge j \in c_x \wedge j \neq k]]\}$ where I is the Set Index. C'' is all combinations in C' that contain j.
  2. $A''_x = \max(A'')$ where A'' are the areas in A that correspond to C''. Find x such that A''_x is Max in A''_x.
  3. $inputs = inputs \bigcup C''_x$ inputs is the union of existing inputs and C''_x

  endwhile

3.4 Maximum polytope (MaxP)

As with MAT the Maximum polytope method is also based on the assumption that mutually distant points offer more information gain to each other. This method takes a more direct approach however by approximating the total enclosed area of all the points involved. Areas are calculated using the Dual Reduced Newton (DRN) algorithm [8] to find the Maximum Volume Ellipsoid (MVE). MVE algorithms are no strangers to being used as subroutines, however they are mostly used for very large examples where the number of points is much greater than the number of dimensions. Here the largest number of points used is 7, whereas the smallest number of dimensions is 6 for classification and 3 for regression.

The DRN Algorithm operates on a matrix of weights, W, formed from a combination of centroids. Section 2.5 details the issues associated with the preconditions of the DRN Algorithm. If the preconditions for the weight matrix are not met then a hyperplane must be found in which they are, to do this the matrix is scaled using Multi-dimensional scaling (Mds)[11]. The matrix is scaled down until the rank of

[W'; $\bar{e}$] is equal to or less than n'+1, where W' is the newly scaled matrix of W and n' is the new dimension.

Note that simply scaling down to F-1 dimensions is not always sufficient to satisfy the preconditions of the DRN algorithm since it is possible for a data set to contain attributes that are entirely empty. Also note that to preserve as much of the matrix W as possible and avoid cumulative scaling errors, W' was not rescaled to W" if the first rescaling was not sufficient. That is, if scaling W down to F-1 dimensions was not sufficient to satisfy the conditions of the DRN algorithm, the original matrix W was then scaled down the F-2 dimensions rather than rescaling W'.

Select the combination of F centroids, which contains centroid k, with the largest enclosed area.

- Generate all combinations C. Where $|C| = \binom{K}{F}$
- For each combination c_x in C

  W =Weight matrix of c_x

  $W' = W$

  $Dim = F - 1$ Dim is the desired dimension.

  where $\bar{e}$ is a row vector of 1s of the appropriate size, and n is the current dimensions of W'.

  while rank $\left(\begin{vmatrix} W' \\ \bar{e} \end{vmatrix} < n + 1 \right)$

  1. $D_{ij} = \|W_i - W_j\|$ $i, j = 1 \rightarrow F$ where D is the distance matrix of W
  2. $W' = Mds(D, Dim)$
  3. $Dim = Dim - 1$

  endwhile

  $A_x = DRN(W')$

- For each expert k to K

  1. $C' = \{c_x | \forall x[x \in I \wedge k \in c_x]\}$ where I is the Set Index. C' is all combinations in C such that all combinations in C' contain k.
  2. $A'_x = \max(A')$ where A' are the areas in A that correspond to C'. Find x such that A_x is max in A'
  3. $inputs = c_x$ inputs is the combination that corresponds to A_x.

3.5 Normalized Triangle (NT)

The Normalized Triangle method focuses more on the mutual distance between points than MAT or MaxP. In this method the actual distances are less important than relative distances, and the enclosed area is never calculated. Triangles are used again in this method because they are easily manipulated despite the number of dimensions. The actual length of each side is calculated using the Euclidean distance between points. The value of the largest side is selected (in case of a tie one is picked arbitrarily) and used to normalize all sides of the triangle. Normalization is applied so that larger triangles do not over shadow smaller ones by virtue of large perimeters. The neighborhood chosen for centroid k is the normalized triangle, which contains centroid k as one of its vertexes and whose sides sum closest to three.

The triangle whose sides sum closest to three represents the triangle whose sides are nearly the same length, an equilateral triangle. The largest any side can be is 1 so the sum can be no greater than three, and the smallest sum could only be two, in which case there is not a triangle but a line (one vertex is a point along that line). If this method were to be expanded to larger number of points, then the criteria would be the combination of centroids whose normalized sides sum closest to the number of edges. If the desired neighborhood size is F then the resulting number of edges E would be:

$$E = \frac{F^2 - F}{2}$$

Select the combination of three centroids, which contains centroid i, such that the formed triangle most closely resembles an equilateral triangle. Since each side of an equilateral triangle are the same size, if every side is normalized in proportion to the largest side, the sum of the sides would be 3.

- Generate all combinations C. Where $|C| = \binom{K}{3}$
- For each combination c_x in C

  $side_1 = ||c_{x1} - c_{x2}||$

  $side_2 = ||c_{x2} - c_{x3}||$

  $side_3 = ||c_{x3} - c_{x1}||$

  $\overline{side} = \overline{side} / \max(\overline{side})$

$sums_x = \sum_{j=1}^{3} side_j$

- For each expert k to K

  1. $C' = \{c_x | \forall x[x \in I \wedge k \in c_x]\}$ where I is the Set Index. C' is all combinations in C such that all combinations in C' contain k.
  2. $sums'_x = \max(sums')$ where $sums'$ are the values in $sums$ that correspond to C'. Find x such that s_x is max in $\overline{sums}$
  3. $inputs = c'_x$ inputs is the combination c'_x which contains centroid i, such that $sums_x$ is closest to 3

Chapter 4

Experimental Procedure

4.1 Parameters

The learning rate was fixed at $\eta = 0.01$, and the momentum constant fixed at $\tau = 0.1$. The SOM size was 8x9 with a total of 72 Nodes and trained for 2500 epochs with a goal of 0.02.

The training set size N was approximately 75% of the whole set size S, more accurately it was N = floor(S-0.25*S). The test set consisted of the remaining S-N examples. Both sets had the ratio of class 1 and class 0 preserved (for classification). This was accomplished as follows: the examples were sorted by their class then were selected with a uniform probability model until 25% of each class has been chosen. These examples were made into the test set, and the remaining examples into the learning set.

The SOM was trained on the learning set, using its full dimensions. That is, it used the examples inputs and outputs as dimensions for the SOM. This was done to aid in clustering the SOM, to further separate examples with similar inputs but different outputs.

Training was performed for 50 iterations, in each training phase examples were presented in random order, and the whole test set is used in each testing phase (not randomly selected). In each iteration the Mean Squared Error (MSE) is recorded for the training and test sets. At the end of training, for classification problems, the MoE is tested for false positives, false negatives, and overall error rates using the test set. This was repeated for 50 trials, with random initializations for weight matrices each time. Since there are no random factors in the selection of clusters, the clustering functions are not recalculated for each trail, meaning the same neighborhood is used for every trial.

4.2 Classification Data sets

This experiment was run using four benchmark data sets detailed in Table 4.1. Imbalanced sets were corrected using a cost-modifying method[3]. To apply imbalance correction the number of Class 0 and Class 1 examples from the learning set was determined, the larger of the two was designated as the Large class and the other as the Small class. The error correction value ec was set at $ec = 1$ for examples whose desired output is of the Large class, and as the ratio of the Large to Small class sizes otherwise giving some value greater than 1. This means that if the ratio of Large to Small is 3:1 the cost of misclassifying the small class becomes 3 times that of misclassifying the large class. The ratio of class 0 to class 1 imbalance is indicated in the Training Imbalance column in Table 4.1.

Data Set	Inputs	Size	Test Set Size	Training Imbalance
BREAST	9	699	175	343:181
C-HEART	13	303	75	122:105
IONO	34	351	88	93:170
LIVER	6	345	86	150:109

Table 4.1: Benchmark Data Sets

The BREAST set is a large benchmark data set obtained from the University of Wisconsin Hospitals. It consists of nine continuous attributes, which describe the characteristics of cells, and two output classes, benign and malignant. The set is fairly easy to classify, although the number of class 0 examples (benign) are nearly twice that of the class 1 examples (malignant).

The C-HEART set consists of the results from patient physicals. The input attributes are a subset of 75 possible attributes, and the output indicates the absence (0) or presence (1) of heart disease. C-HEART is a more typical data set that is of moderate difficulty and the imbalance in the set is only marginal.

The Space Group at John Hopkins University collected readings from their radar antennas and compiled them into the IONO benchmark data set. The object of the data set is to classify the radar returns from the ionosphere. The IONO set is considered more difficult due to the large number of dimensions and the number of class 1 examples (good) are nearly twice that of the class 0 examples (bad).

In the BUPA Medical Research Ltd. LIVER disorder data set each instance represents a male individual. The input attributes represent blood tests, which are thought to be sensitive to liver disorders that arise from excessive drinking, and also number of drinks per day. The outputs indicate the absence (0) or presence (1) of a liver disorder. This set is considered a difficult set in which to classify despite the low dimensions and only slight imbalance. this may be due to the fact that the data set does not have easily identified clusters.

4.3 Regression problems

The regression data was run with two chaotic time series functions Model IV [7] and the Lorenz function [10] (x value prediction only). Test sets were generated by breaking the data set into five equal segments and selecting 25% of each segment with uniform random probability. Before applying the SOM the learning data was normalized in each dimension according to the following form: $d_{ij} = \frac{d_{ij} - \bar{d}_j}{\sigma}$, where d_{ij} is the i^{th} example in the j^{th} dimension. The test data was also normalized in each dimension, using the $\bar{d}$ and σ parameters of the learning set.

Since the output ranges were not [0, 1] a transfer function other than sigmoid was necessary, in this case a simple linear function was used. Also the posterior probability model had to be changed to a regression model:

$$h_i = \frac{g_i * \exp(-\frac{1}{2}||d - y_i||^2)}{\sum_{j=1}^{K} g_j * \exp(-\frac{1}{2}||d - y_j||^2)} \tag{4.1}$$

$$\bar{h} = \frac{\bar{h}}{\sum_{i=1}^{K} h_i} \tag{4.2}$$

Table 4.2: Regression Data Sets

Data Set	Size	Dimensions	Test Set Size
Lorenz	940	3	235
Model IV	801	5	200

The Lorenz equation [10] yields a chaotic time series, the equations are:
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(r - z) - y$
$\dot{z} = xy - bz$

The constants σ, r, and b are initialized to 10, 28, and 8/3 respectively. The current data set was generated using numeric integration with a time range of [0, 15] and initial values of x, y and z as 10. The data set inputs are x, y and z values, and the desired outputs are the x values for the next time step.

The Model IV function [7] yields a wave that is regular at first but becomes unstable in later stages. The output function of the model is:

$$y_p(k+1) = f[y_p(k), y_p(k-1), y_p(k-2), u(k), u(k-1)]$$

where

$$f[x_1, x_2, x_3, x_4, x_5] = \frac{x_1 x_2 x_3 x_5 (x_3 - 1) + x_4}{1 + x_3^2 + x_2^2}$$

and

$$u(k) = \begin{cases} \sin(\frac{2\pi k}{250}) & k \leqslant 500 \\ 0.8\sin(\frac{2\pi k}{250}) + 0.2\sin(\frac{2\pi k}{25}) & k > 500 \end{cases}$$

The inputs for the data set are the values of x_1 to x_5 and the desired outputs are the values of y, which range [-1, 1], for all values of $k = 0$ to 800.

4.4 Experimental Design

Three experiments were conducted. The initial test was a detailed look at the performance of the architectures in which the number of clusters was arbitrarily set at 15, and the number of neighbors was 3. Second the number of clusters was varied from 5 to 40 by increments of 5 with the number of neighbors fixed at 3. And Third the neighborhood size was varied from 3 to 7 with the number of clusters fixed at 15. Based on the results of the first and second experiments, not all architectures were examined in the final experiment. The MoEII architecture was not used because it does not have a neighborhood size, and the Normalized Triangle was not used because it's performance was neither robust nor competitive with the other architectures.

The criteria for a good architecture were that it must perform significantly better than MoEII, it must be competitive with the architecture that performs best, it must not have misleading MSE values, and finally it must be robust. Originally this did not include KNN either, but it was later added due to its performance on the Regression sets.

Chapter 5

Results

Significance testing was conducted via t-test at the 95% confidence interval ($p \leq 0.05$). The primary focus of this paper has been to compare the accuracy of differing architectures, therefore four accuracy measurements have been taken. The first measurement has been Mean Squared Error (MSE) which is a commonly used statistic for numeric prediction. Equation 5.1 defines the computation of MSE values, where n is the sample size, $x_1 ... x_n$ are actual results, and $d_1 ... d_n$ are the desired results.

$$\frac{(x_1 - d_1)^2 + (x_2 - d_2)^2 + ... + (x_n - d_n)^2}{n} \tag{5.1}$$

Because mean squared error is sensitive to large error values and outliers, and because it can hide poor performance on smaller classes three additional metrics were used to verify the results of MSE. Mean false positives, false negatives and total error rates are expressed as percentages. The false positive rate (FPR) is defined as the proportion of class 0 examples that are wrong (equation 5.2), and false negative rate (FNR) is defined as the proportion of class 1 examples that are wrong (equation 5.3). The overall or total error rate is simply the proportion of all examples that are incorrect (equation 5.4).

$$FPR = \frac{\text{\# of incorrect class 0 examples}}{\text{total \# of class 0 examples}} \tag{5.2}$$

$$FNR = \frac{\text{\# of incorrect class 1 examples}}{\text{total \# of class 1 examples}} \tag{5.3}$$

$$Total\ Error = \frac{\text{\# of incorrect examples}}{\text{total \# of examples}} \tag{5.4}$$

False positives and false negatives allow us to verify that apparently good performance is not merely the result of good perforance on a single class but poor performance on another. This is often the case when a data set has a sizable imbalance between the classes, there can be 100% success on the large class, 100% error on the small class and still have a reasonably small MSE value.

In the following section results are divided first by data set (classification first followed by regression sets), and then by experiment. Results are first compared to the MoEII architecture as a basis of performance, and then to each other. Even though the networks were not run to convergence, comparisons to standard algorithms are also discussed.

5.1 Classification

5.1.1 BREAST

Details of the BREAST data set are discussed in Section 4.2. Some previous results from standard classifiers are presented in Table 5.1. The values were originally presented as a percentage of correctly classified examples, the values here are percent error instead.

Table 5.1: BREAST Set Previous Results

Algorithm	Percent Error
2-Hyperplanes	6.5
3-Hyperplanes	4.1
1-NN	6.3

5.1.1.1 Initial Tests

All architectures show significantly lower MSE values than the MoEII architecture (Fig 5.1). However KFN, MAT, and MaxP had the lowest scoring MSE values, performing significantly better than both KNN and NT. The three architectures with the lowest error have tighter bounds than the others. Since MSE values are sensitive to large values in the results, the wider ranges could indicate that the other architectures have more inconsistent values or are more prone to outliers. This is especially apparent in the MoEII and NT plots whose upper whiskers show a skewed distribution. KFN, MAT, and MaxP on the other hand have tight bounds where extreme values are easily labeled as outliers.

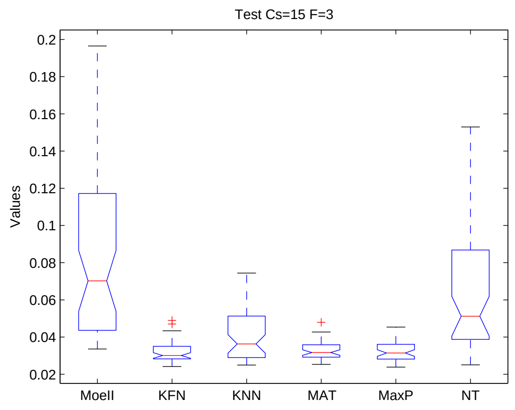


Figure 5.1: Box-plot of Test Minimums for BREAST data set

For False Positives (Fig 5.2) all architectures, except NT, showed significantly lower FPR values than the MoEII architecture. KFN, KNN, MAT, and MaxP were statistically indistinguishable from each other, and all performed significantly lower than NT. The former three have extremely tight ranges and although the Maximum Polytope architecture was not significantly different from the others, it did have a slightly larger range. This shows a higher consistency of False Positive values with KFN, KNN, and MAT.

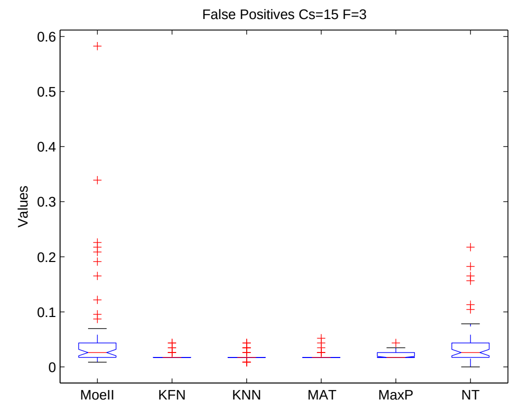


Figure 5.2: Box-plot of False Positives for BREAST data set

False Negatives (Fig 5.3) showed a wider range of values, indicating that it was here that contributed to most of the error of the architectures. Although none of the architectures performed significantly different than the standard MoEII architecture, KFN, MAT and MaxP have significantly lower FNR values than KNN and NT. K-Furthest Neighbors, Maximum Area Triangle and Maximum Polytope still have tighter ranges of values then the other architectures. K-Nearest Neighbors fails to do so here and this is where it falls behind the others.

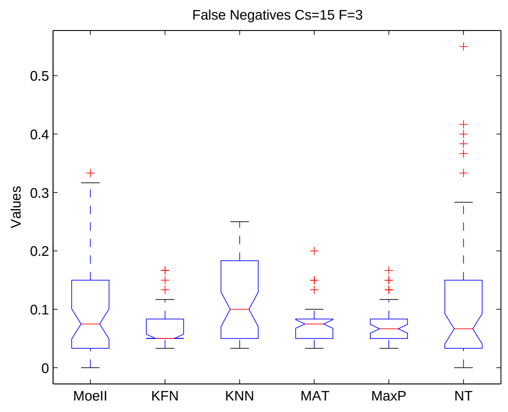


Figure 5.3: Box-plot of False Negatives for BREAST data set

Fig 5.4 shows the Total Error rates for the BREAST set. Most error rates of the architectures are consistent with the MSE values and show significantly lower Error rates than MoEII, except NT. For all three error rate metrics the Normalized Triangle architecture failed to show any significant difference from MoEII, yet it did show a significant difference for MSE values. This is most likely a fluke of the data, in the error rate metrics the significance test was more apt to consider higher data points of MoEII as outliers which brings the mean down closer to the mean of NT. Looking at the range of values alone, we can see that NT is not comparable with the best performing architectures.

As with the false negatives KFN, MAT, and MaxP also performed significantly lower than KNN and NT. The K-Nearest Neighbors architecture's poor performance on the False Negative Rates factored in heavily to the over all error rate. We can see that its high MSE value is not the result of a few outliers but rather consistent misses on one class. On the other hand the three best performing architectures had consistently good performance for all metrics

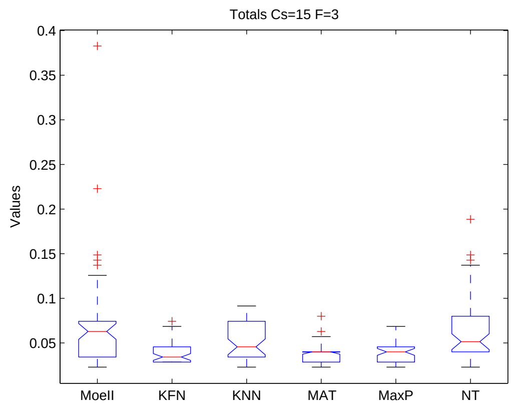


Figure 5.4: Box-plot of Total Error Rates for BREAST data set

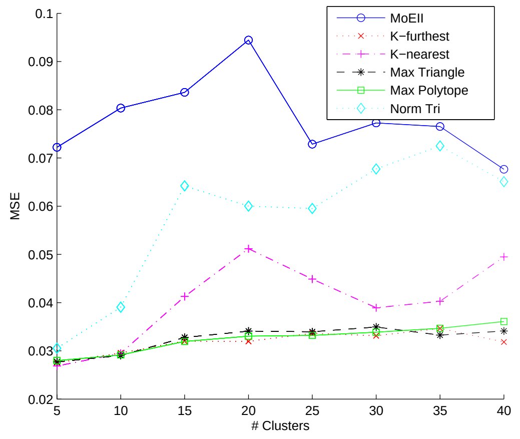
The mean error rates for False Positives, False Negatives, and Total Error rates are presented in Table 5.2. Even though the training of the architectures was not performed to convergence we can see that the best performing architectures have superior performance to those presented in Table 5.1. Even the worst performing of the new architectures are at least comparable to the previous ones.

Table 5.2: Error Rates for BREAST set

Architecture	FPR	FNR	Total Errors
MoEII	6.38	8.70	7.18
K-Furthest Neighbors	2.05	7.17	3.81
K-Nearest Neighbors	1.91	11.70	5.27
Maximum Area Triangle	1.98	7.53	3.89
Maximum Polytope	2.05	7.30	3.85
Normalized Triangle	4.00	11.63	6.62

5.1.1.2 Number of Clusters

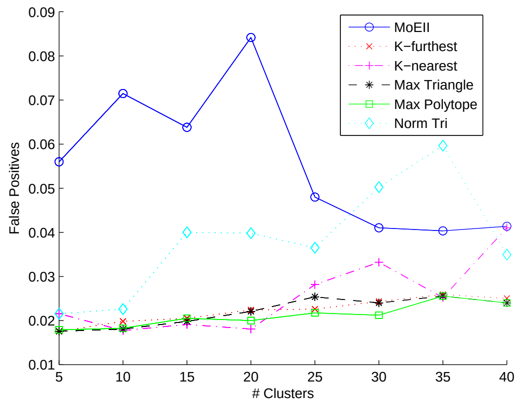
All architectures show significantly lower MSE values than MoEII for all number of clusters, except for NT which fails to show significantly lower MSE values for 30 to 40 clusters. We can see from Fig 5.5 that K-Furthest Neighbors, Maximum Area Triangle and Maximum Polytope have steady MSE values that only gradually increase with the number of clusters. The others have sharp increases and decreases throughout the domain.



(a) Test Mins

Figure 5.5: Plot of Test mins

For False Positives (Fig 5.6) KFN, MAT and MaxP showed significantly lower FPR values for all number of clusters except for 35 clusters. They also show the same gradual increase in values as the number of clusters increases. KNN only showed significantly lower values for 5 to 20 clusters. The reason for this can be seen in Fig 5.6 where K-Nearest Neighbors follows the same trends as the previous three architectures until it becomes more erratic after 20 clusters. The Normalized Triangle architecture only showed significantly lower FPR values than MoEII for 5, 10, and 20 clusters, after which its values were close to if not more than those of MoEII.

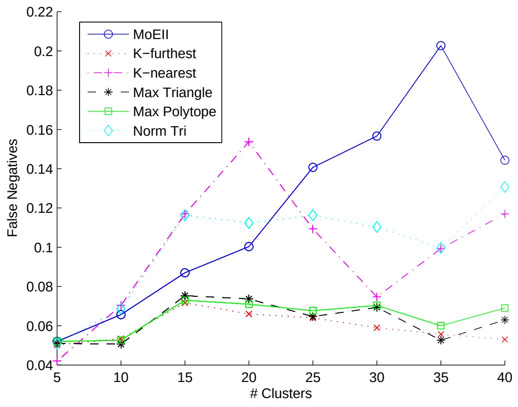


(a) False Positives

Figure 5.6: Plot of False Positive Rates

False Negatives show larger values over all, and also show the only sharp increase in the KFN, MAT and MaxP architectures. Interestingly the increase is at 15 clusters, where the initial tests were conducted. Because there are no sharp decreases in FNR values afterwards, it doesn't seem likely that there was an error in collecting the data at 15 clusters. Rather it seems that the optimal number of clusters maybe 5 or 10. Because of MoEIIs initially low False Negative rates KFN, MAT and MaxP only had significantly lower FNR values after 20 clusters.

The False Negative values for KNN show a rapid increase which leads to a significantly higher FNR value for 20 clusters. It only has significantly lower values for 30 and 35 clusters. NT only manages to show a significantly lower value for 35 clusters, when MoEII peaks. It shows fairly consistent values from 15 to 35 clusters, already indicating that the later growth in MSE values after 25 clusters is a result of the False Positive values.



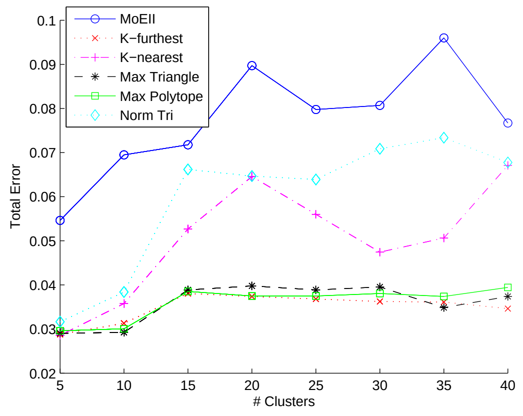
(a) False Negatives

Figure 5.7: Plot of False Negative Rates

As a result of poor performance at the beginning of False positives and at the end of False Negatives, the total error rates for MoEII are high for all numbers of clusters (Fig 5.8). If not for the peak at 20 clusters (a result of false positives) and the peak at 35 clusters (a result of false negatives) then MoEII would show a clear gradual increase in error rates as the number of clusters increase. This is contrary to what the MSE values show, which is a gradual decrease with number of clusters. Apparently the MSE values are not heavily influenced by the false negative values this is further shown by the following architectures.

KFN, MAT and MaxP had significantly lower error rates than MoEII for all number of clusters. The increase in false negative error rates from 10 to 15 clusters shows itself here as well. Since the increase is not as visible in the MSE values we see again that the False Negative errors contribute only modestly to mean squared error. This is a result of the imbalance in the set, since FNR values represent the smaller class 1. The number of clusters that was best for these architectures were 5 or 10 clusters. Increasing the number of clusters beyond 15 did not show noticeable changes in the total error rates.

KNN only failed to show significantly lower error rates for 40 clusters, but it still showed worsening performance as the number of clusters increased. NT showed significantly lower errors for 5, 10, 20, and 35 clusters. These two architectures are more consistent with their MSE values.

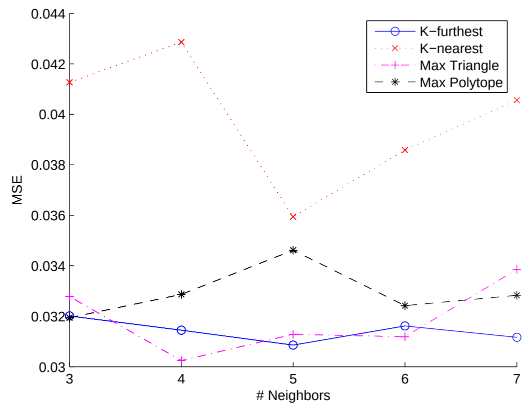


(a) Total Error

Figure 5.8: Plot of Total Error Rates.

5.1.1.3 Higher Neighborhood sizes

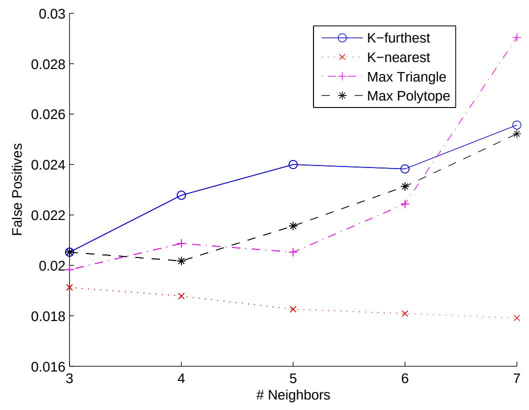
Fig 5.9 displays the mean MSE values for the best performing architectures from the previous two sections for higher neighborhood sizes. We see that K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polytope have very similar values. However MaxP has significantly higher MSE values than MAT on F=4, and significantly higher values than both KFN and MAT for F=5. There were no significant differences between KFN and MAT, although KFN seemed to have less fluctuation. KNN showed significantly higher MSE values than the others for all values of F, except for F=5 where it dips and the Maximum Polytope architecture peaks. There were no obvious changes in the MSE values as the neighborhood sizes increased.



(a) Test Mins

Figure 5.9: Plot of Test mins

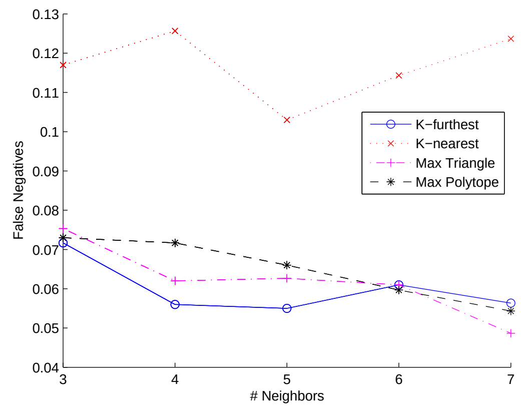
The False Positives (Fig 5.10) . Show an increasing trend for K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polytope, where as K-Nearest Neighbors has a gradually decreasing FPR. MAT performs significantly lower than KFN for F=5. KNN performs significantly better than KFN for F=4 onward, and better than MAT, and MaxP for F=5 onward. With a large neighborhood size KNN is a superior architecture for classifying False Positives.



(a) False Positives

Figure 5.10: Plot of False Positive Rates

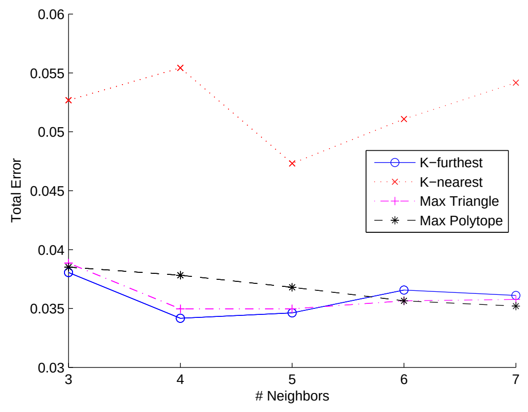
With the False Negatives the performance seem to be the reverse of the False Positives. The False Negatives of KFN, MAT and MaxP gradually improve with the neighborhood sizes. Their performance is very similar but there are some points where KFN performs significantly lower than MaxP for F=4 and 5. KNN performs significantly worse than the other architectures for all values of neighborhood sizes, which is similar to what we see in the MSE values.



(a) False Negatives

Figure 5.11: Plot of False Negative Rates

The False Negatives have the largest impact on the Total Errors, as can be seen in Fig 5.12. The only difference between the best architectures is where KFN performs significantly lower than MaxP for F=4. KNN followed the same trends that it did for MSE values and False Negatives, its performance was significantly worse than the other architectures across the entire domain. KNNs good performance on False Positive Rates is clearly over shadowed by its FNR values. The Total Error rate seems to heavily weigh in the False Negatives as we have seen in the varying cluster sizes. However this time the False Negatives and False Positives share their contributions to the MSE values.



(a) Total Error

Figure 5.12: Plot of Total Error Rates.

5.1.2 C-HEART

Details of the C-HEART data set are discussed in Section 4.2. Some previous results from standard classifiers are presented in Table 5.3. The values were originally presented as a percentage of correctly classified examples, the values here are percent error instead.

Table 5.3: C-HEART Set Previous Results

Algorithm	Percent Error
Bayes	62.6
KNN	52.2
Kohonen	30.7
C5.0	25
Cn2	23.3
C4.5	21.9
NewID	15.6

5.1.2.1 Initial Tests

All architectures show significantly lower MSE values than the MoEII architecture (Fig 5.13). MAT and MaxP also perform significantly lower than NT but no other differences can be seen. KFN, MAT and MaxP do have an advantage of having smaller data ranges than the others.

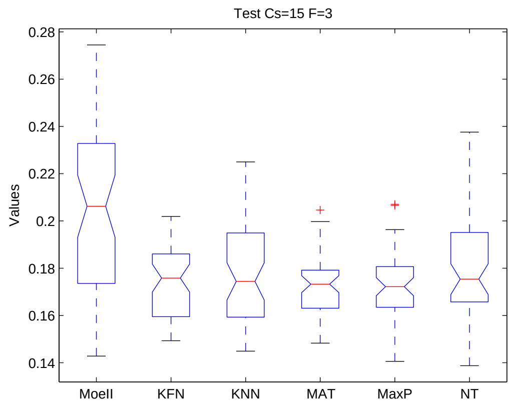


Figure 5.13: Box-plot of Test Minimums for C-HEART data set

Figures 5.14,5.15, and5.16 show the False Positives, False Negatives and Total Error rates respectively for C-HEART. For each metric all architectures have significantly lower values than the MoEII architecture, but show no significant difference from each other. While this is good for the new architectures, it makes it difficult to determine which of the new architectures are most promising. We can see that KFN, MAT and MaxP tend to have tighter limits on their range of data, especially for False Positives and False Negatives. This is consistent with the results seen in their MSE values. On this basis we could say that they are more consistent with their performance, but this cannot be statistically stated.

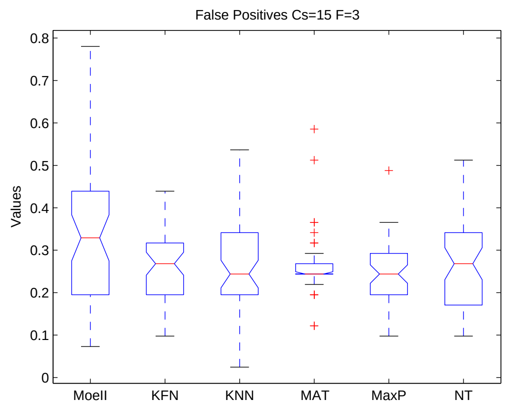


Figure 5.14: Box-plot of False Positives for C-HEART data set

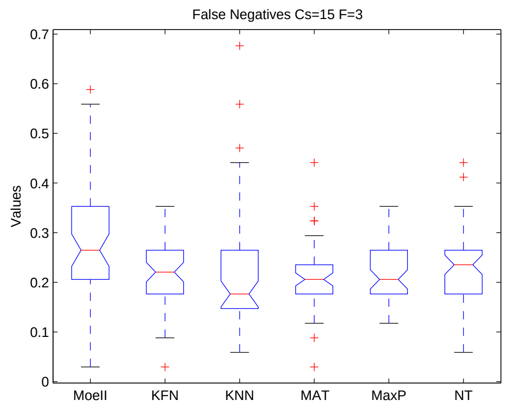


Figure 5.15: Box-plot of False Negatives for C-HEART data set

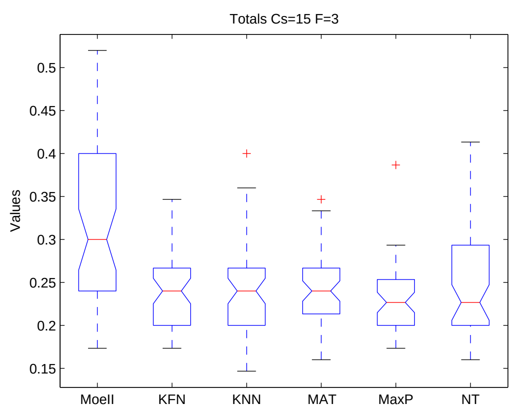


Figure 5.16: Box-plot of Total Error for C-HEART data set

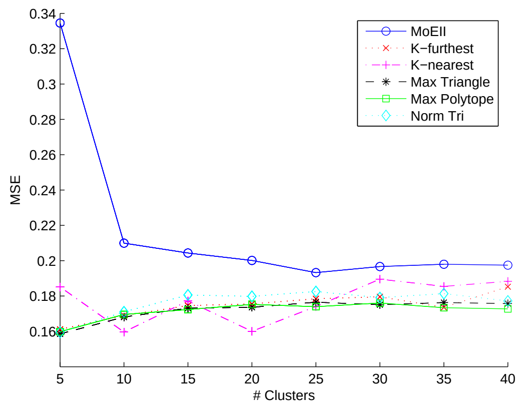
The C-HEART data set has had many algorithms tested on it, and therefore there is a plenty of material to compare performance on. Table 5.4 shows that the new architectures do not out perform the best existing algorithm but they are comparable to Cn2 and C4.5 (Table 5.3). Their performance would likely have been better if they were trained from more than 50 iterations.

Table 5.4: Error Rates for C-HEART set

Architecture	FPR	FNR	Total Errors
MoEII	35.66	27.71	32.05
K-Furthest Neighbors	25.95	21.47	23.92
K-Nearest Neighbors	25.27	22.71	24.11
Maximum Area Triangle	26.44	21.24	24.08
Maximum Polytope	24.39	21.82	23.23
Normalized Triangle	26.63	22.88	24.93

5.1.2.2 Number of Clusters

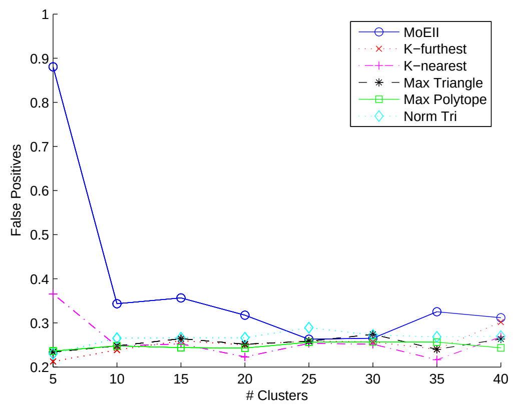
The only point at which an architecture does not have significantly lower MSE values than the MoEII architecture is K-Nearest Neighbors for 30 and 40 clusters. Just as we have seen in the initial tests the new architectures have very similar performance across the number of clusters. In Fig 5.17 we can see that the new architectures only increase slightly with the number of clusters. MoEII on the other hand has extremely poor performance when the number of clusters are very small. In the initial tests we saw that KNN had larger data ranges than some of the other architectures (Fig 5.13), this wider range may account for the fluctuation in values seen in Fig 5.17.



(a) Test Mins

Figure 5.17: Plot of Test mins.

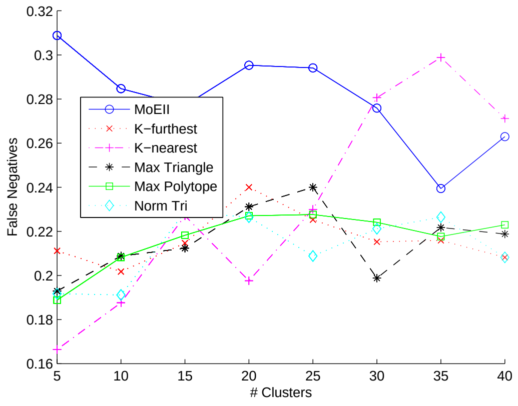
K-Furthest Neighbors, K-Nearest Neighbors and Maximum Area Triangle had significantly lower FPR values than MoEII except where their values raise at 40 clusters and where the False Positive values of MoEII dip from 25 to 30 clusters (Fig 5.18). MaxP did not have a rise at 40 clusters so it remained significantly different at point, otherwise it had similar performance to the others. NT only showed significantly lower values for 5 to 20 clusters.



(a) False Positives

Figure 5.18: Plot of False Positive Rates.

In Fig 5.19 we see that MoEII has a more gradual improvement than in False Positives or MSE values. For False Negatives KFN shows significantly lower False Negative rates for all number of clusters. In Fig 5.19 the K-Nearest Neighbors architecture shows a clear increase in False Negative rates as the number of clusters increases, thus it only has significantly lower FNR values than MoEII for up to 25 clusters. After 25 clusters it's FNR values are above those of MoEII, in fact KNN even shows a significantly higher value when it peaks at 35 clusters. MAT and MaxP showed significantly lower FNR values for up to 30 clusters. Surprisingly NT performed significantly better than MoEII for all number of clusters except 35 clusters. The Normalized Triangle architecture has never shown particularly good performance, at best it has been competitive but rarely superior to MoEII.



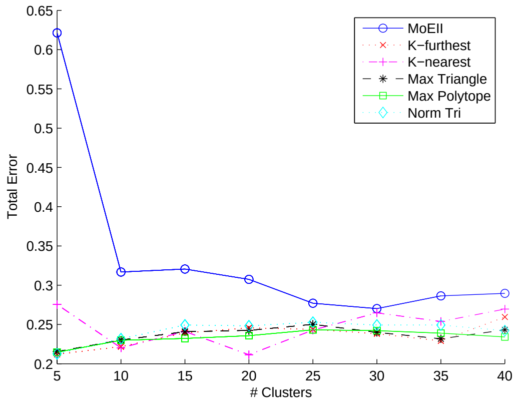
(a) False Negatives

Figure 5.19: Plot of a) Test mins b) False Positive Rates c) False Negative Rates and d) Total Error Rates.

The MSE values and the Total Errors rates (Fig 5.20) resemble each other very much, and therefore the MSE values are a good representation of the percent errors. The K-Furthest Neighbors, Maximum Area Triangle and Maximum Polytope architectures showed significantly lower error rates for all number of clusters. They remained consistent, showing the same trends in all metrics.

K-Nearest Neighbors on the other hand showed fluctuating errors in False Positives (Fig 5.18) that did not clearly increase or decrease, it also showed increasing errors in False Negatives (Fig 5.19). As a result, while the total error rates fluctuate a bit they also show a gradual increase in error as the number of clusters increase (Fig 5.20). KNN showed significantly lower error rates for 5 to 25 and 35 clusters.

The only time the Normalized Triangle architecture does not have significantly lower error rates is when MoEII dips from 25 to 30 clusters. The total error rates show a slightly smaller disparity between MoEII and NT than the MSE values do. Apparently this is a result of its performance on False Negatives.



(a) Total Error

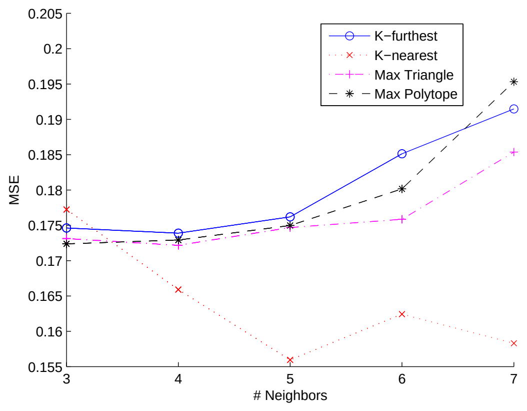
Figure 5.20: Plot of Total Error Rates.

Over all the effect of increasing the number of clusters had a positive effect on MoEII. With more specialized experts it was able to segment and classify the problem more effectively. It had a somewhat negative effect on the K-Nearest Neighbors architecture though due to the worsening performance on Class 1 examples.

Increasing the number of clusters splits existing larger clusters, these splits would take place where there are high concentrations of data which are themselves potential clusters. High concentrations of data rarely happen on the edges of the input space, clusters that do occur here are those most often picked by KFN, MAT and MaxP due to the fact that they are distant centroids that would contribute to a large area if chosen. The highest concentrations of data tend to occur near the middle of the input space, where KNN makes most of its choices. So where the previous three are not making many new choices, the K-Nearest Neighbors architecture is. This is a problem for KNN if the new choices offer less information then the larger landmarks.

5.1.2.3 Higher Neighborhood sizes

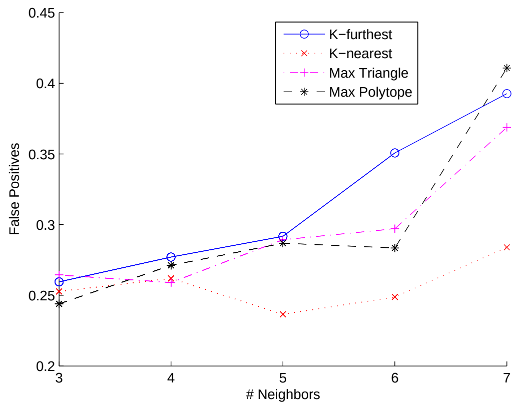
As the Neighborhood sizes increased the K-Nearest Neighbors architecture had decreasing MSE values. In Fig 5.21 we can see that it performed significantly better than KFN and MAT from F=4 onward, and better than MaxP from F=5 onward. The other architectures had no significant differences between each other, and their MSE values increased noticeably with the neighborhood sizes.



(a) Test Mins

Figure 5.21: Plot of Test mins.

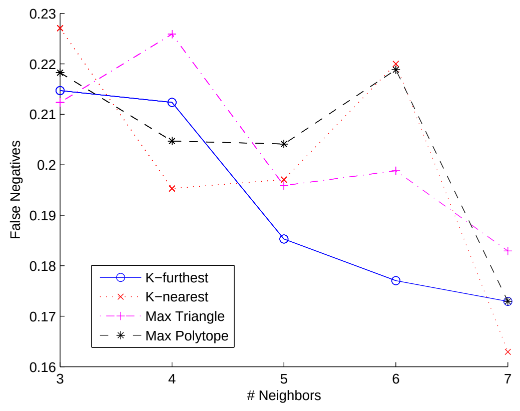
The False Positive rates of the architectures resemble the MSE values. KNN outperforms the KFN, MAT, and MaxP architectures in the later neighborhood sizes, having significantly lower False Positive rates than the others for F=5 onward, only failing to show a significantly lower value than Maximum Polytope for a neighborhood size of 6. The only significant difference among the others is where MaxP performs significantly lower than KFN for F=6.



(a) False Positives

Figure 5.22: Plot of False Positive Rates.

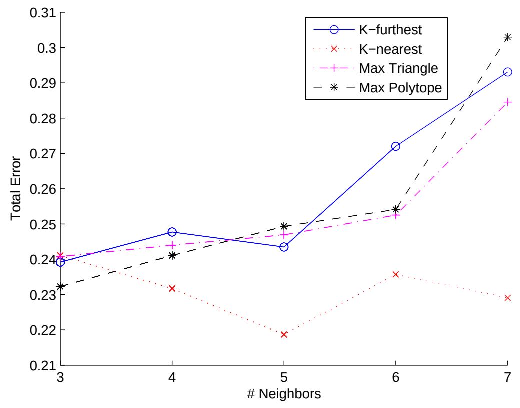
In Fig 5.23 the False Negatives show a different trend, they all seem to improve with the neighborhood size. None of the architectures stand out from the others except where KFN performs significantly lower than MaxP and KNN for F=6. At the same neighborhood size the Maximum Area Triangle architecture also performed significantly better than KNN.



(a) False Negatives

Figure 5.23: Plot of False Negative Rates.

The improvements in False Negatives are very small however and have little impact on the Total Errors (Fig 5.24). Because the changes in False Positive rates were so large they dominate the Total Error rate and also the MSE values. The K-Nearest Neighbors architecture had significantly lower error rates than the other architectures from F=5 onward. But because of its rise at F=6, KNN did not show a significantly lower error rate than MAT and MaxP for that instance. KNNs Total Error rates did not show an obvious improvement as neighborhood sizes increased as compared to its MSE values, so it is difficult to conclude weather there is really an improvement or not.



(a) Total Error

Figure 5.24: Plot of Total Error Rates.

At the lowest neighborhood sizes the architectures were competitive, but as the neighborhood sizes increased there was a divergence as K-Nearest Neighbors remained around the same level (Total Errors) while the others had larger and larger errors. This could indicate two things: first that the next nearest neighbors to a cluster offer very little information gain. Second, by increasing the neighborhood sizes we are forcing K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polytope to select centroids that are suboptimal or misleading.

5.1.3 IONO

Details of the C-HEART data set are discussed in Section 4.2. Some previous results from standard classifiers are presented in Table 5.5. The values were originally presented as a percentage of correctly classified examples, the values here are percent error instead.

Table 5.5: IONO Set Previous Results

Algorithm	Percent Error
Linear perceptron	9.3
Non-linear perceptron	8
1-NN	7.9
C4	6
Back propagation	4
IB3	3.3

5.1.3.1 Initial Tests

All architectures have mean MSE values that are significantly less than that of the MoEII architecture (Fig 5.25). In addition the Maximum Area Triangle and Maximum Polytope architectures both performed significantly better than the others, but with no difference from each other. The range of values seen by their whiskers in Fig 5.25 are somewhat smaller than the others. This indicates that the good performance of the two is consistent across trials.

Of the new architectures KNN had the largest range, comparable to that of MoEII. It also had the largest MSE value of the new architectures.

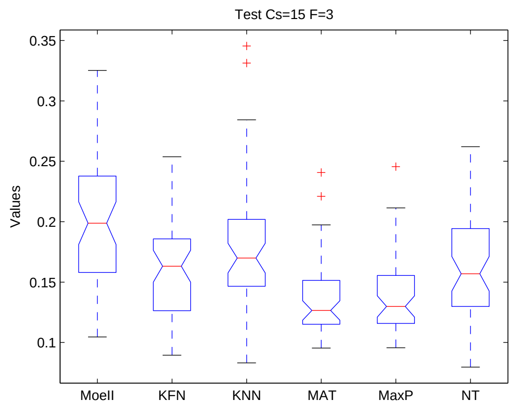


Figure 5.25: Box-plot of Test Minimums for IONO data set

Similar to the MSE values, all of the new architectures have significantly lower False Positive values than the MoEII architecture (Fig 5.26). The only significant difference between the new architectures is where the Maximum Area Triangle architecture performs significantly better than the K-Furthest Neighbors architecture, otherwise the new architectures are statistically indistinguishable.

It should also be noted that MAT, MaxP, and NT have tight ranges and discard a number of points as outliers. To discard data points that close to the mean when MoEII and KNN kept points in the same range, means that the data in the three architectures are more strongly correlated.

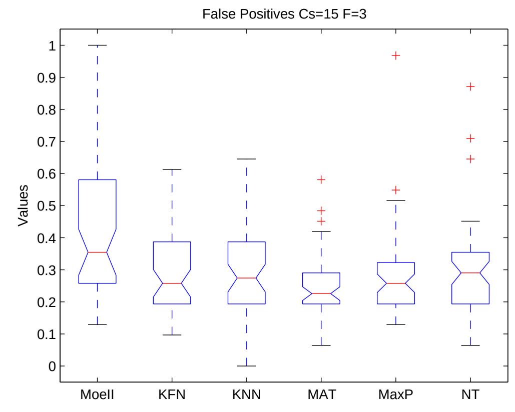


Figure 5.26: Box-plot of False Positives for IONO data set

Fig 5.27 shows us that the Maximum Area Triangle and Maximum Polytope architectures have the best performance again. This time, however, they are the only architectures to show significantly lower False Negative rates than MoEII. They also performed significantly better than KFN, KNN and NT, but there was no significant difference from each other.

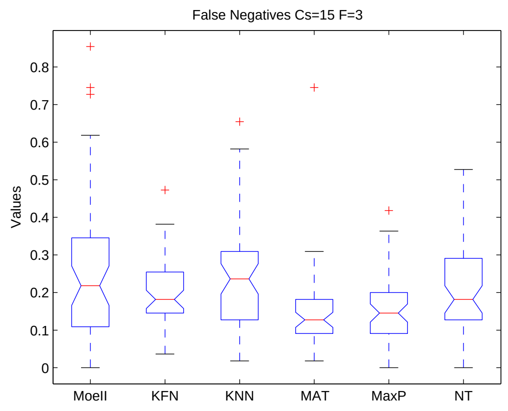


Figure 5.27: Box-plot of False Negatives for IONO data set

The Total Error rates (Fig 5.28) follow the same pattern as the False Positive rates and MSE values. All architectures show significantly lower error rates than MoEII. And in addition MAT and MaxP performed significantly better than the K-Furthest Neighbors, K-Nearest Neighbors and Normalized Triangle architectures.

The improvement seen in the new architectures, over MoEII, seems to come mostly from their superior performance on False Positive rates. Only MAT and MaxP had improved performance on False Negatives and False Positives, hence the reason that their over all performances are best.

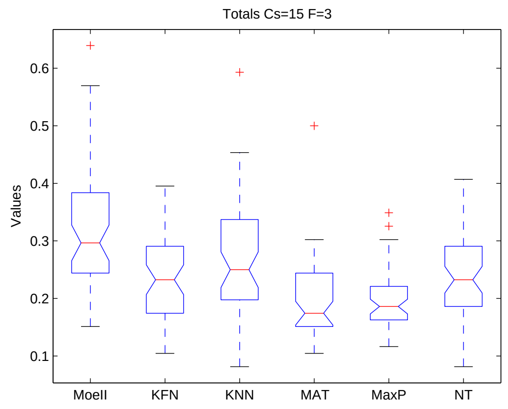


Figure 5.28: Box-plot of Total Error rates for IONO data set

As we can see in Table 5.6 the new architectures do not seem to competitive with existing algorithms at all. The best of the new architectures (MAT) has an average error that is twice that of the worst algorithm from Table 5.5 (a Linear perceptron). As stated before the results presented here have not been run until the convergence of the networks. It may still be possible that the new architectures are at least competitive with existing methods.

Table 5.6: Mean Error Rates for IONO set

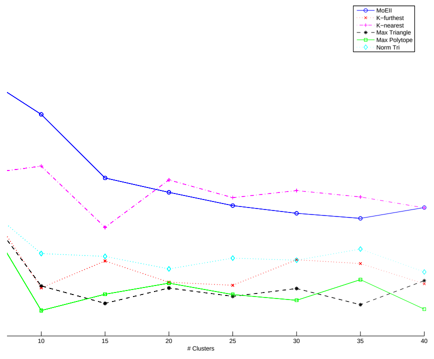
Architecture	FPR	FNR	Total Errors
MoEII	42.39	25.93	31.86
K-Furthest Neighbors	30.52	20.00	23.79
K-Nearest Neighbors	29.94	23.56	25.86
Maximum Area Triangle	26.13	15.67	19.44
Maximum Polytope	28.71	15.49	20.26
Normalized Triangle	29.48	20.18	23.53

5.1.3.2 Number of Clusters

We can see from Fig 5.29 that K-Furthest Neighbors, Maximum Area Triangle and Maximum Polytope architectures have the best performance of the new architectures. They showed significantly lower MSE values, than MoEII, for all number of clusters.

The K-Nearest Neighbors architecture had the poorest performance having only showed significantly lower MSE values for up to 15 clusters. After 15 clusters it even had MSE values that were higher than MoEII. Across all number of clusters it had significantly higher MSE values than the other new architectures.

The Normalized Triangle architecture also had significantly lower MSE values than MoEII and KNN for all number of clusters. However it periodically had significantly higher MSE values than MAT and MaxP.

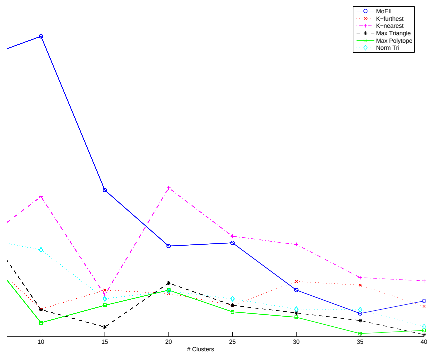


(a) Test Mins

Figure 5.29: Plot of Test mins.

Increasing the number of clusters did not seem to have a large impact on the MSE values. Only from 5 to 10 clusters did there seem to be an improvement. Despite some minor fluctuations they stayed at about the same values. This is different form the other architectures where there would be a gradual increase in error for some of the new architectures.

In Fig 5.30 we can see that MoEII begins at a high False Positive rate an quickly descends as the number of clusters increases. This decent is much more pronounced than in Fig 5.29. All of the new architectures have significantly lower FPR values than MoEII up until 15 clusters. After 15 clusters KFN, MAT, MaxP, and NT have one more significantly lower value at 25 clusters, when they all drop slightly and MoEII increases slightly. Also after 15 clusters, the K-Nearest Neighbors architecture has higher values than MoEII (although this is not statistically shown).



(a) False Positives

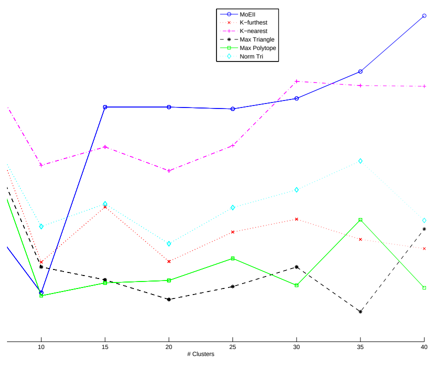
Figure 5.30: Plot of False Positive Rates.

The False Negatives in Fig 5.31 are an interesting case. Usually MoEII begins with a much higher error rate than the other architectures at smaller number of clusters. Here MoEII begins with a significantly lower FNR value than the others, however it quickly rises.

The K-Nearest Neighbors architecture never has significantly lower values than MoEII. After 15 clusters all of the other architectures have lower values than MoEII. For KFN they are significantly lower for 20 to 40 clusters. MAT and MaxP are the same except they include 15 clusters as significant. The Normalized Triangle is the same as KFN except that it fails to show a significant difference at 30 clusters.

Among the new architectures the Maximum Area Triangle and Maximum Polytope architectures had significantly lower False Negatives than KNN for all number of clusters. The K-Furthest Neighbors architecture, also out performed KNN, although it failed to show its FNR values as significantly lower than KNN for 5 and 15 clusters.

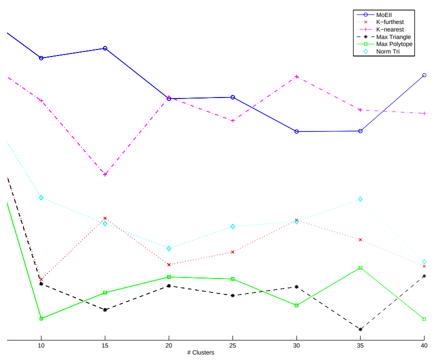
As with the MSE values, the architectures do not show a clear increase or decrease in performance after 5 clusters, except MoEII and KNN which have increasing error rates.



(a) False Negatives

Figure 5.31: Plot of False Negative Rates.

For the Total Error rates all architectures show significantly lower values than MoEII for all number of clusters, except for the K-Nearest Neighbor architecture which shows no significant differences from MoEII at all(Fig 5.32). We can see the influence of False Negatives in the Total Error rates. MoEII has a bowl shape, a result of the combination of FPR and FNR values. The False Negatives drive up the initial values of the other architectures. False Positives curb the fluctuations in False Negatives, resulting in smoother lines. The MSE values of the new architectures follow the Total error rates rather closely. The MSE values of MoEII are more representative of the False Positives.



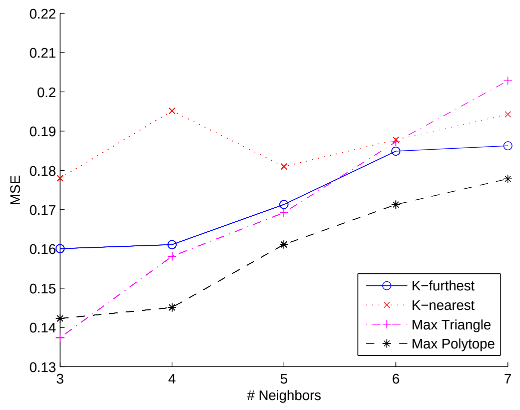
(a) Total Error

Figure 5.32: Plot of Total Error Rates.

5.1.3.3 Higher Neighborhood sizes

As the neighborhood sizes increased the K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polygon architectures had increasing MSE values. The MSE values of the K-Nearest Neighbors architecture remained about the same as neighborhood sizes increased, however it's initial values were larger than the other architectures. That is why MAT had a significantly lower MSE value at F=3, KFN had a lower value at F=4, and MaxP had significantly lower MSE values for neighborhood sizes of both 3 and 4.

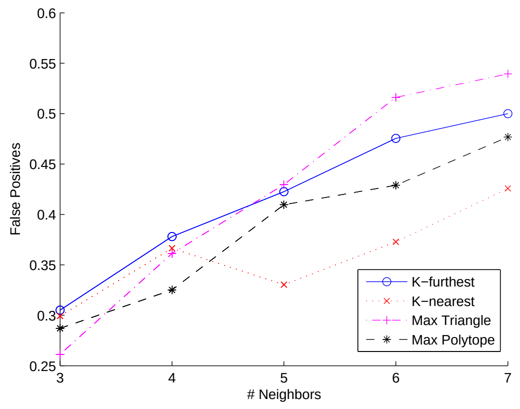
MAT and MaxP both performed best in the initial tests, having significantly lower MSE values than even KFN. However MaxP seems to have the advantage as the neighborhood sizes increase. It has the lowest vales from F=4 onward, although it is not shown significantly. Of the best performing architectures it is the least sensitive to the neighborhood sizes.



(a) Test Mins

Figure 5.33: Plot of Test mins.

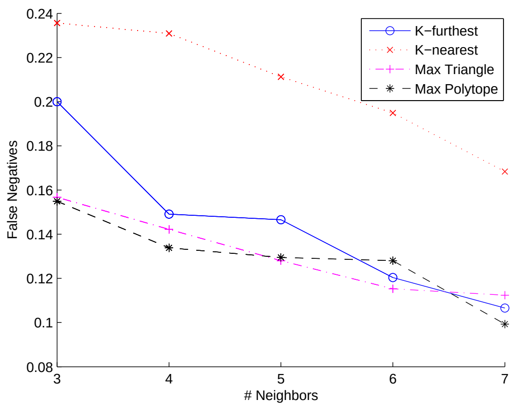
The False Positives are the only point in the IONO data set that we see the K-Nearest Neighbors architecture out perform KFN or MAT (Fig 5.30). KNN showed significantly lower FPR values than KFN for neighborhood size 5, and lower than the Maximum Area Triangle architecture for neighborhood sizes of 6 and 7. It never showed any significantly lower False Positives than the Maximum Polytope however.



(a) False Positives

Figure 5.34: Plot of False Positive Rates.

Compared to the other architectures KNN had very poor False Negative values. The Maximum Area Triangle and Maximum Polytope architectures out performed KNN for all neighborhood sizes. As did the K-Furthest Neighbors architecture except on neighborhood size 3. Among KFN, MAT, and MaxP, there were no significant differences except where MAT and MaxP out performed KFN on the initial tests (F=3).

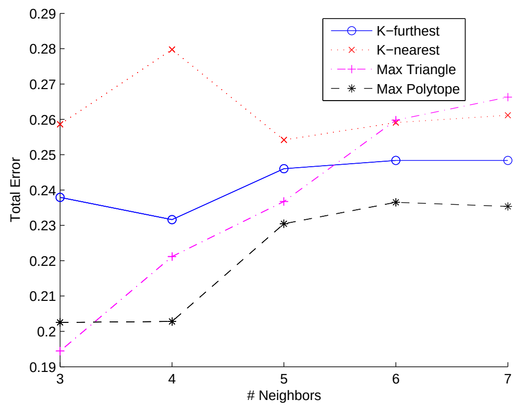


(a) False Negatives

Figure 5.35: Plot of False Negative Rates.

Comparing Fig 5.30 and Fig 5.31 we see that increasing the neighborhood sizes increases False Positive rates and decreases False Negative Rates. The improved FNR values are not large enough to offset the FPR values. And thus in Fig 5.32 we see increasing error rates for KFN, MAT, and MaxP. On the other hand, K-Nearest Neighbors has a gradual enough decrease in FPR values that the improved False Negatives are able to maintain a relatively stable Total Error rate, the error rate at F=4 being an exception.

Because of the KNNs hight initial values. MAT and MaxP out perform it for neighborhood sizes of 3 and 4, and KFN out performs it for only F=4. The K-Furthest Neighbors architecture does not perform as well as the previous two. It has significantly higher error rates than both MAT and MaxP at neighborhood size of 3, and higher than just MaxP for F=4.



(a) Total Error

Figure 5.36: Plot of Total Error Rates.

5.1.4 LIVER

Details of the LIVER data set are discussed in Section 4.2. Some previous results from standard classifiers are presented in Table 5.7. The values were originally presented as a percentage of correctly classified examples, the values here are percent error instead.

Table 5.7: LIVER Set Previous Results

Algorithm	Percent Error
C5.0	34.9
Tree	34
Blk L-GP	33.3
PG L-GP	32.7

5.1.4.1 Initial Tests

Most of the new architectures had significantly lower MSE values than the MoEII architecture, they include: KFN, KNN and MaxP. However, as can be seen in Fig 5.37, the performance of the K-Nearest Neighbors architecture was superior to the others by a sizable margin. The range of values for KNN was comparable to that of the other architectures, so it is unlikely that its mean MSE value is the result of a few outliers.

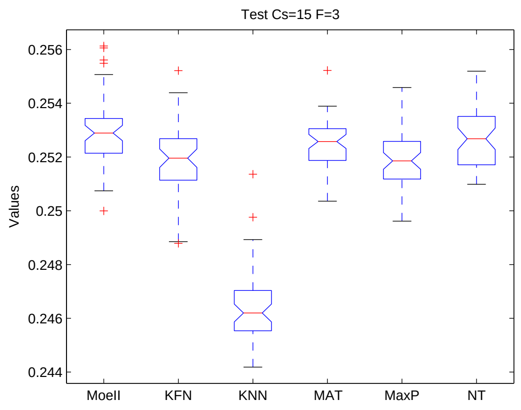


Figure 5.37: Box-plot of Test Minimums for LIVER data set

For False Positives only the K-Nearest Neighbors architecture had significantly lower False Positive rates than the MoEII architecture. The other architectures had significantly higher FPR values. Their ranges were very tight, and even their outliers were not comparable to KNN.

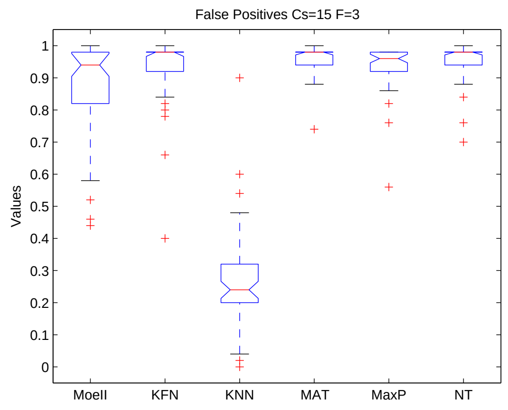


Figure 5.38: Box-plot of False Positives for LIVER data set

The situation is almost completely reversed for the False Negative rates. KNN performs significantly worse than the rest of the architectures, while the rest perform significantly better significantly better than MoEII. This is a situation normally seen in heavily unbalanced data sets. To minimize MSE values, or some other metric, the learning algorithm summarily classifies every thing as the larger class. The result is 100% classification for the large class and 0% classification for the small class.

While the large class in the LIVER set is approximately 1.5 times the size of the smaller class, precautions have been taken to correct imbalance. If this were the case however then we should see extremely small MSE values in Fig 5.37, but we do not.

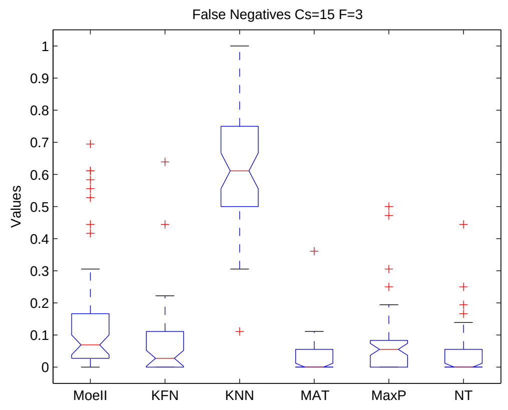


Figure 5.39: Box-plot of False Negatives for LIVER data set

As expected the Total Errors are a blend of the False Positives and False Negatives Fig 5.40, resulting in all architectures having Total Error rates that are in the mid range. KNN still performs significantly better than all of the other architectures with an average Total Error rate that is less than 50%. Error rates were included to back up the MSE values, in case they were misleading. In this case it is the Total Error rates that were misleading.

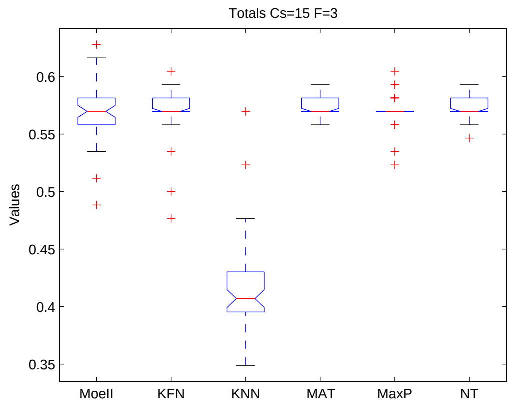


Figure 5.40: Box-plot of Total Errors for LIVER data set

Table 5.8 shows the mean Total Error rates, as well as the mean False Positive and False Negative Rates. Even the K-Nearest Neighbors architecture was not comparable to the previous findings in Table 5.7. Perhaps with more training the new architectures could be competitive, however considering the gap it is unlikely.

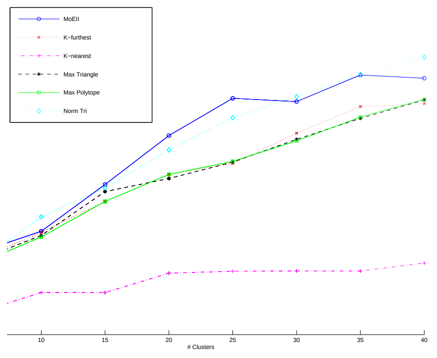
Table 5.8: Error Rates for LIVER set

Architecture	FPR	FNR	Total Errors
MoEII	86.84	15.33	56.91
K-Furthest Neighbors	92.60	7.50	56.98
K-Nearest Neighbors	27.00	61.39	41.40
Maximum Area Triangle	96.20	3.50	57.40
Maximum Polytope	92.68	7.61	57.07
Normalized Triangle	95.16	4.39	57.16

5.1.4.2 Number of Clusters

In Fig 5.41 we can see that the MSE values of the architectures fell into three trends. The fastest growing of the trends included MoEII and the Normalized Triangle architecture. This was followed by KFN, MAT, and MaxP. The slowest growing trend was the K-Nearest Neighbors architecture.

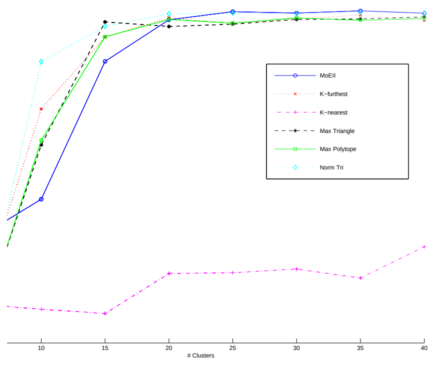
The architectures in the second trend had significantly lower MSE values than MoEII from 20 clusters onward. KNN had significantly lower MSE values than MoEII and the rest of the architectures for all number of clusters.



(a) Test Mins

Figure 5.41: Plot of Test mins.

The only time that KFN, MAT, MaxP and NT had significantly lower False Positive rates than MoEII was at 5 clusters. It was also at this point that K-Furthest Neighbors was competitive with K-Nearest Neighbors, and MAT and MaxP had significantly better performance than KNN. Unfortunately their MSE values grew rapidly and KNN had significantly better performance than all architectures.

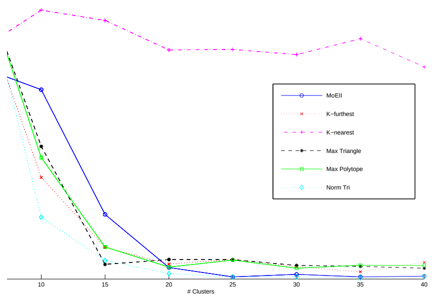


(a) False Positives

Figure 5.42: Plot of False Positive Rates.

As with in the initial tests, the False Negative results were the inverse of the False Positives. KFN, MAT, MaxP and NT had a significantly higher FNR value than MoEII for 5 clusters, then briefly had lower values. From 25 clusters onward the higher False Negatives of KFN, MAT, and MaxP were significant.

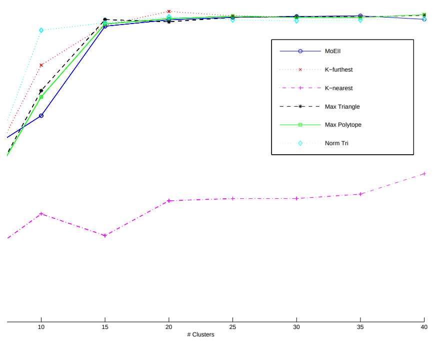
KNN had significantly higher False Negatives than the other architectures for 10 clusters onward.



(a) False Negatives

Figure 5.43: Plot of False Negative Rates.

With Total Errors we see the same blending process as before. But we can see clearly that increasing the number of clusters has detrimental effect on all architectures. At its minimum point, the K-Nearest Neighbors architecture may have been competitive with the results in Table 5.8, if more training were applied.



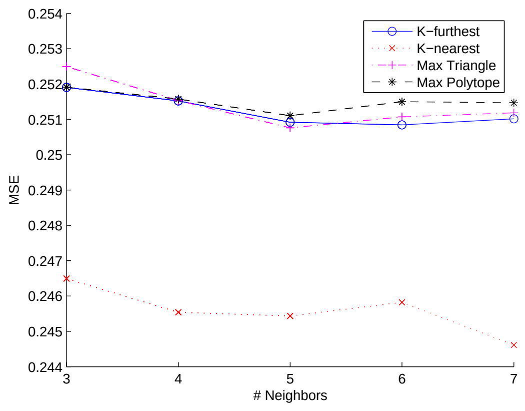
(a) Total Error

Figure 5.44: Plot of Total Error Rates.

It is apparent from increasing the number of clusters, that the poor performance of the architectures is not a result of insufficient clusters to represent the data set. In [3] it was found that the complexity of a problem can cause a disparity in false positives and false negatives such as that seen in imbalanced sets. So it is likely that the liver set contains many clusters of classes that are heavily intermixed, and the architectures presented here are not prepared to deal with this.

5.1.4.3 Higher Neighborhood sizes

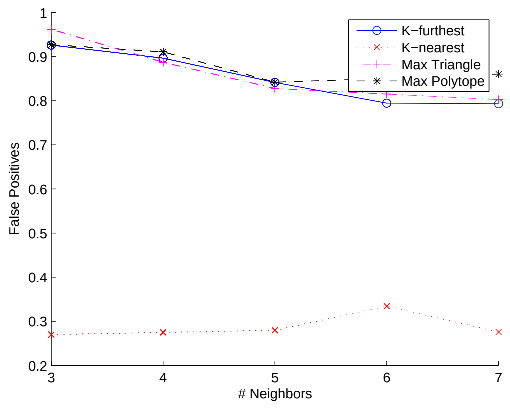
Looking at Fig 5.45 it seems that the architectures are not very sensitive to the number of neighbors. But there is a slight decrease in MSE values and the neighborhood size increased. There are a few significant differences between KFN, MAT, and MaxP architectures but nothing to extent of the K-Nearest Neighbor architecture. KNN had significantly lower MSE values for all neighborhood sizes.



(a) Test Mins

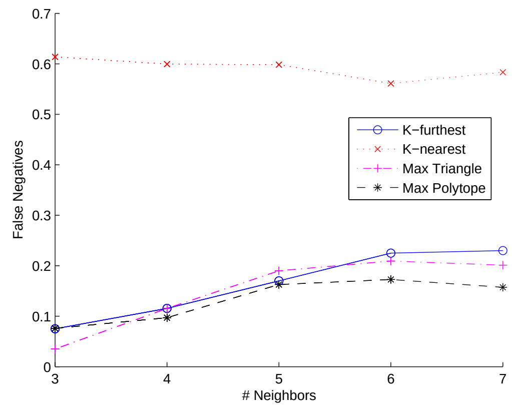
Figure 5.45: Plot of Test mins.

The False Positives closely resembled the MSEs in Fig 5.46 where the K-Nearest Neighbors architecture performed best. KNN seemed insensitive to changing neighborhood sizes. Aside from a few significant differences the other architectures had very similar values, and improved slightly with neighborhood size. The False Negative values were an inverse of the False Positives (Fig 5.47).



(a) False Positives

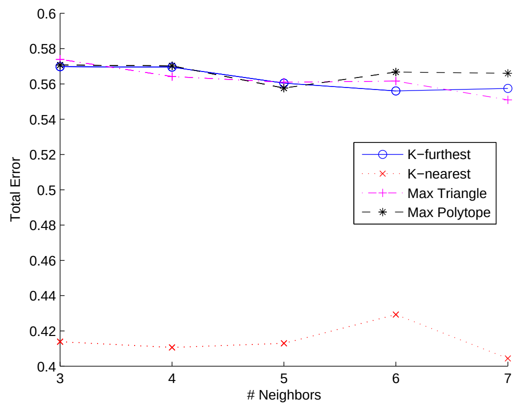
Figure 5.46: Plot of False Positive Rates.



(a) False Negatives

Figure 5.47: Plot of False Negative Rates.

The Total Errors show KNN to be less sensitive to neighborhood size then the MSE values in Fig 5.45. The other architectures still show a small improvement with changing neighborhood sizes.



(a) Total Error

Figure 5.48: Plot of Total Error Rates.

5.2 Regression

5.2.1 Lorenz

5.2.1.1 Initial Tests

All of the new architectures show significantly lower MSE values than the MoEII architecture except for MAT. From Fig 5.49 we see that the Maximum Area Triangle method has a large number of outliers, this extends the normal range of the data and prevents a significant difference from being found. The range of values for K-Nearest Neighbors and Normalized Triangle are very small, but only KNN has significantly better performance than KFN, MAT and MaxP. This is very different from what we have seen in classification problems, both KNN and NT normally have a much larger range of values.

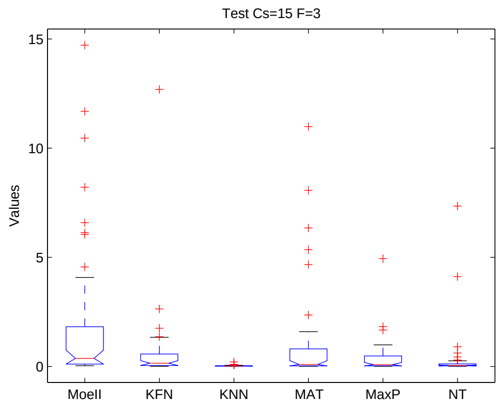


Figure 5.49: Box-plot of Test Minimums for Lorenz data set

5.2.1.2 Number of Clusters

The MoEII architecture had extremely high MSE values at lower number of clusters but quickly began to improve then reach a minima (Fig 5.50). Because of MoEIIs improvement across the domain and the growing values of the K-Furthest Neighbor and the Maximum Polytope architectures, they had significantly lower MSE values then the MoEII architecture for 5 to 15 clusters, and it had significantly higher values for 30 to 40 clusters.

The K-Nearest Neighbors architecture had significantly lower MSE values than MoEII for all number of clusters, except for 35 clusters where they come very close to each other. Increasing the number of clusters had little to no effect on the KNN architecture.

The Maximum Area Triangle architecture had similar performance to KFN and MaxP, it had steadily growing MSE values as the number of clusters increased. Because of this it only had significantly lower values than MoEII for 5 and 10 clusters and then it had significantly higher values for 30 to 40 clusters. The Normalized Triangle architecture had significantly lower MSE values all the way up to 25 clusters. but because its MSE values grew with number of clusters, although not as much as KFN, MAT, and MaxP, it had a significantly higher value at 35 clusters.

Additionally, KNN had significantly lower MSE values than KFN, MAT, and MaxP for all number of clusters. It was even significantly lower than NT for 20 clusters and 30 to 40 clusters. NT had significantly lower error values than KFN, MAT and MaxP from 20 clusters onward, although it failed to show significantly lower values than MaxP for 20 and 35 clusters.

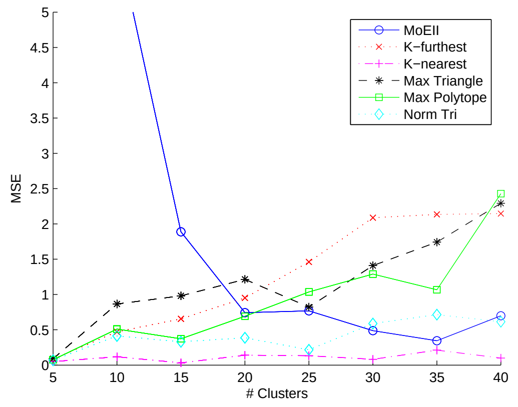


Figure 5.50: Plot of Test Minimums for Lorenz data set

5.2.1.3 Higher Neighborhood sizes

In Fig 5.51, we see that increasing the Neighborhood sizes increases the MSE values of the K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polytope architectures. These architectures have no significant differences from each other except where KFN and MaxP both perform significantly lower than MAT for F=7. The K-Nearest Neighbor architecture performed significantly better than all of the other architectures for all neighborhood sizes. It also showed very little increase in MSE values as the neighborhood sizes increased.

Not only is the K-Nearest Neighbor architecture insensitive to number of clusters but it is also insensitive to neighborhood sizes, for the Lorenz series. Because it performs well for all values, we could create a network with minimum values of each. This would give us an uncomplicated network that is fast running and quick to train.

5.2.2 Model IV

5.2.2.1 Initial Tests

Similar to the MSE values of the Lorenz tests, KNN performed significantly better than all architectures except for the Normalized Triangle architecture. There were no

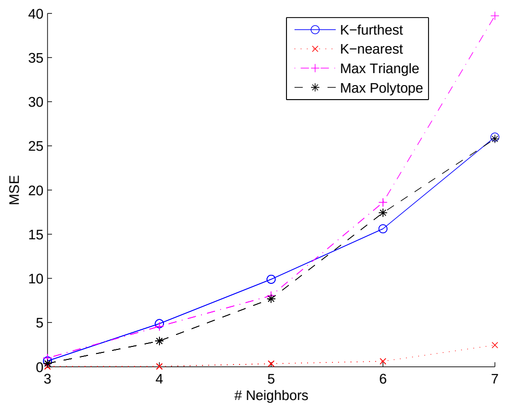


Figure 5.51: Plot of Test Minimums for Lorenz data set

significant differences among the rest of the architectures. We can see from Fig 5.52 that the K-Furthest Neighbors architecture has the largest range of values, followed by MoEII, MaxP and then MAT. In the classification problems KFN, MaxP, and MAT had the tightest ranges and they certainly didn't have ranges that were comparable to MoEII. These three architectures seem unsuited to regression problems.

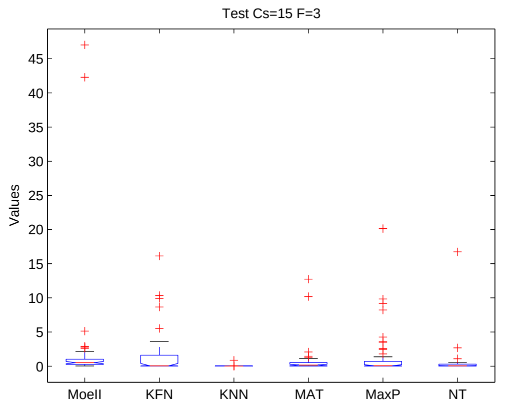


Figure 5.52: Box-plot of Test Minimums for Model IV data set

5.2.2.2 Number of Clusters

The K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polytope architectures each only had one significantly lower MSE value than MoEII, at 5 clusters (Fig 5.53), for 20 clusters and onward they all had significantly higher values. Looking at Fig 5.53 MoEIIs performance at 15 clusters seems anomalous, although the Normalized Triangle architecture peaked at that point as well. From Fig 5.52 we can tell that this is the result of the two highest outliers in the data.

KNN showed significantly lower MSE values than the MoEII architecture for all number of clusters. Its performance became significantly better than KFN and MAT from 15 clusters onward, and better than MaxP from 20 clusters onward. The K-Nearest Neighbors performance is especially noteworthy since the standard deviations remain small across all number of clusters [ref to appendix: means and stds].

The Normalized Triangle architecture was comparable to the KNN architecture. There were a few points were it did not perform quite as well, such as where it raises at 30 and 35 clusters. At these same points NT fails to have significantly lower MSE values than MoEII. It also performs well compared to the other architectures as well, after 15 clusters its MSE values are all significantly lower than KFN, MAT, and MaxP. Even before that, at 5 and 10 clusters, its values were significantly lower. However, the Normal Triangle architecture does not perform significantly better than the other architectures as often as the K-Nearest Neighbor architecture does.

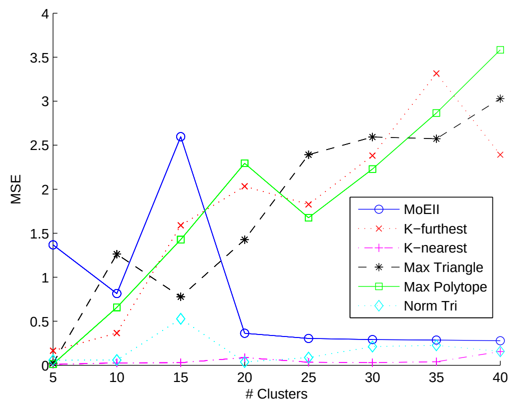


Figure 5.53: Plot of Test Minimums for Model IV data set

5.2.2.3 Higher Neighborhood sizes

In Fig 5.54 we see that increase the neighborhood sizes has the same effect on the architectures as increasing the number of clusters. KFN, MAT, and MaxP have increasing MSE values as neighborhood sizes increase, where as KNN has continuously low values. The K-Furthest Neighbor architecture showed significantly lower MSE values than the other architectures for all neighborhood sizes. Also there was little to no fluctuation, where as it had subtle rises and falls as seen in Fig 5.53.

Among the other architectures there were some differences. The K-Furthest Neighbor architecture showed significantly lower MSE values than MaxP for neighborhood sizes after four. MAT only had a significant difference at F=6, where it could be shown that it had significantly lower error than MaxP. No other differences could be shown to be statistically significant, although from Fig 5.54 we can see that MaxP had the worst performance.

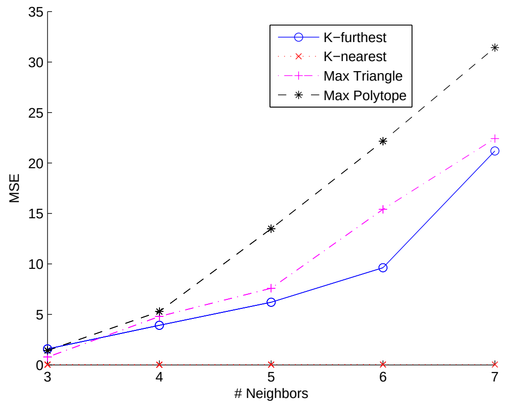


Figure 5.54: Plot of Test Minimums for Model IV data set

Chapter 6

Conclusion

For the initial tests the best performing architectures are given in Table 6.1. For classification problems it seems that the K-Furthest Neighbors, Maximum Area Triangle and Maximum Polytope architectures are best with the later two being slightly more robust. This is not true however for the LIVER data set in which K-Nearest Neighbors is the clear winner. KNN was also the best architecture for regression problems. The Normalized Triangle architecture was also a good choice for regression but it did not seem to perform quite as well as KNN.

The architectures that select distant centroids are a good choice for generic classification problems. However they have poor performance on problems that are are more complex, like the LIVER set and regression problems. It seems that choosing closer centroids offer the most information gain in those circumstances since the conditions change so rapidly. If we take the example of a wave, it would be more benefical to take the nearest neighbors because they will offer use more information about how the function is changing at that point in time. Distant neighbors will give us very little local informaiton.

For most of the new architectures increasing the number of clusters has at best no effect on performance. It can even create small increases in error. The only

Table 6.1: Initial Test Winners

Data Set	Architecture(s)
BREAST	KFN, MAT, MaxP
C-HEART	KFN, MAT, MaxP
IONO	MAT, MaxP
LIVER	KNN
Lorenz	KNN
Model IV	KNN

architecture that is improved by increasing the number of clusters is MoEII. Most of the other architectures have their minimum error at very small number of Clusters.

This is good because the cost of generating the architectures depends heavily on the number of clusters or experts. This is espeically true for the Maximum Area Triangle and Maximum Polytope architectures. Reducing the number of clusters and experts also as the effect of speeding up training, because there are fewer weight matricies to update.

Increasing the number of clusters for regression problems was especially bad for KFN, MAT, and MaxP.

It was expected that increasing the neighborhood size would improve the perfomance of the architectures. However, it made things worse for KFN, MAT, and MaxP for most circumstances. Only for the LIVER set did increasing the neighborhood size improve performance for those architectures.

The improvements of K-Furthest Neighbors, Maximum Area Triangle, and Maximum Polytope with respect to neighborhood size seems to confirm that the poor performance on the set is due to the complexity of the LIVER set. As the architectures choose more neighbors they are getting a more complex view of the data set itself. In less complex data sets adding neighbors hurt performance because the architectures began with the optimal neighbors, selecting more neighbors means listening to suboptimal or misleading inputs. With the LIVER set there is more information gain from new neighbors. At least this is true for KFN, MAT, and MaxP who tend to choose centroids from the outskirts of the input space, depending on the structure of the data set these may be of a kind. On the other hand centroids near the middle of the input space are more likely to change classes rapidly. Thus a method like K-Nearest Neighbors is more likely to include varying classes in its earlier choice of neighbors, giving it a good view of the consistency of the data set to begin with. Adding more neighbors to KNN does not give much new information because we already know what the data set looks like.

<–edit stops here

All of the architectures out perform the MoEII design, improving both MSE values and false report errors. Therefore, not only did the reduction of inputs prove lossless but also more effective. KFN, MAT and MaxP yield the best performance for BREAST, C-HEART, and IONO data sets. Although KNN is competitive for C-HEART, and clearly superior for LIVER it does not seem to be robust for classification problems. Increasing the neighborhood sizes did not show any significant advantages between the architectures, however KFN and MAT were most competitive while MaxP tended to have higher error values. MAT may have been more robust but considering the improvement was not very large the time taken in generating the neighborhood may not be worth the marginal performance gains.

The effect of increasing the neighborhood size had little effect on the BREAST set, but it worsened performance for the C-Heart and IONO sets while improving performance for the LIVER set. Only the dimensions and the complexity of the data varied from data set to data set, and since the dimensions did not correlate to the results shown only the complexity of the data can account for these differences.

On the regression problems KNN performed best, with NT being comparable since it also performed significantly better than the others. When the neighborhood sizes are increased the MSE values of KFN, MAT and MaxP also increase, with MaxP doing exceptionally bad. It may be worthwhile to examine KNN at higher neighborhood sizes and consider it as a regression specific architecture.

K-Furthest Neighbors, Maximum Area Triangle and Maximum Polytope were robust classifiers that performed better than the MoEII architecture. They were all very competitive with each other. The architectures had converse performance on regression problems, K-Nearest Neighbors and Normalized Triangle architectures perform best and the prior three performed poorly.

Bibliography

[1] Tang B. Heywood M. I., Shepherd M. Input partitioning to mixture of experts. pages 227–232. IEEE - INS International Joint Conference on Neural Networks, May 2002.

[2] Jordan M. I. and Jacobs R. A. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[3] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, 6, Nov 2002.

[4] L. Kuncheva. Clustering-and-selection model for classifier combination. volume 1, pages 185–188, Sept 2000.

[5] Chiu S. L. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2:267–278, 1994.

[6] V. Ramamurti and J. Ghosh. Advances in using hierarchical mixture of experts for signal classification. volume 6, pages 3569–3572, May 1996.

[7] Narendra K. S. and Parthasarathy K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[8] P. Sun and R.M. Freund. Computation of minimum-volume covering ellipsoids. *Operatoins Research*, 52(5):690–706, 2004.

[9] Kohonen T. The self-organizing map. *Proceedings of the IEEE*, 78:1464–1480, Sept 1990.

[10] Principe J.C. Wang L., Motter M. A. Local dynamic modeling with self-organizing maps and applications to nonlinear system identifiction and control. *Proceedings of the IEEE*, 86(11):2240–2257, 1998.

[11] Krzanowski W.J. and Marriott F.H.C. *Classification, Covariance Structures and Repeated Measurements*, volume 2. Oxford University Press, 1994.

[12] Geva S. Wong, M. and M. Orlowski. Pattern recognition from neural network with functional dependency preprocessing. volume 1, pages 387–390, May 1997.

[13] Zhang Y. and Gao L. On numerical solution of the maximum volume ellipsoid problem. *SIAM Journal on Optimization*, 14(1):53–76, 2003.

[14] R.R. Yager and D.P. Filev. Approximate clustering via the mountain method. *Systems, Man and Cybernetics, IEEE Transactions on*, 24:1279–1284, Aug 1994.

Appendix A

A chapter in the appendix

Appendix B

The next appendix

Appendix C

Glossary

Attribute Used interchangeably with “feature". See Section **??**.