GP Under Streaming Data Constraints: A Case for Pareto Archiving?

Aaron Atwater[1], Malcolm I. Heywood[1], and A. Nur Zincir-Heywood[1]

[1]*Faculty of Computer Science, Dalhousie University, Halifax, NS. Canada*

Article originally appears at GECCO under ACM copyright 2012
`http://dl.acm.org/citation.cfm?doid=2330163.2330262`

Abstract

Classification as applied to streaming data implies that only a small number of new training instances appear at each generation and are never explicitly reintroduced by the stream. Pareto competitive coevolution provides a potential framework for archiving useful training instances between generations under an archive of finite size. Such a coevolutionary framework is defined for the online evolution of classifiers under genetic programming. Benchmarking is performed under multi-class data sets with class imbalance and training partitions with between 1,000's to 100,000's of instances. The impact of enforcing different constraints for accessing the stream are investigated. The role of online adaptation is explicitly documented and tests made on the relative impact of label error on the quality of streaming classifier results.

1 Introduction

Dynamic environments in general imply that, rather than evolution being performed against a single static data set, evolution is instead conducted relative to a series of samples, themselves subject to some form of variation [13, 1]. As such this might imply that the task for which reward is received varies over time, the behaviour of the environment itself drifts / varies over time (non-stationary) or that fitness evaluation against all training instances is not feasible, thus potentially introducing a learning bias (only part of the underlying task is known at any cycle of evolution).

Various arguments have been made for the potential superiority of evolutionary methods in dynamic environments. Specifically, the case for maintaining a set of diverse models for detecting change within dynamic environments has been made [13]. Likewise, the availability of a diverse set of solutions makes it more likely that a 'good enough' solution may exist at any time to ensure the continued survival of the population [13]. A recent survey article on the open issues in Genetic Programming (GP) re-emphasized the general appropriateness of dynamic environments (Section 2.3 in [15]). This was qualified by remarking that although there has been some success in providing GP solutions to specific applications in which dynamic properties exist (e.g., [4]), there has been little analysis of the behaviour of GP in such environments in general.

In this work, we conduct a benchmarking study under a **streaming data** formulation of the classification task i.e., evolution is limited to a **single pass** through the training data. Unlike the classification task as traditionally assumed, a streaming data constraint implies that the learning algorithm cannot (directly) adopt a batch mode of operation. Specifically, **batch** frameworks for classification assume that any exemplar from the training partition can be accessed at uniform cost. Two forms of fitness evaluation fall under the batch model: either evaluate over all training instances at each generation, or over a subset of training instances as sampled without constraint from the entire training repository at each generation. Conversely, **online** learning as applied to streaming data implies that only when training data explicitly lies within a current 'window' or 'block' of sequential records can it be utilized for training purposes. The content of the block is updated incrementally – as in a sliding window [1] – or under the assumption of non-overlapping updates in which the entire block content is updated [18].

Within the above streaming data context the interest of this study lies in measuring the potential contribution made by competitive coevolutionary Pareto archiving. Such a scheme provides the capability to identify training instances that should be retained (archived) for longer than the current sample of instances provided by a sliding window. Fitness evaluation is only ever conducted over a subset of training instances as defined by the sliding window content and Pareto archive. This is somewhat different to previous research, as we do not explicitly assume that the current content of a sliding window is sufficient to support the identification of appropriate models (e.g., as in [4]) or that coevolving the size of the sliding window is sufficient for retaining useful training instances (e.g., [16]).

2 Classification under streaming data constraints

The task considered in this work represents a base case in a wide range of potential issues associated with machine learning as applied to the analysis of data streams. Thus, of the genres frequently associated with data stream analysis – change detection, online data description, frequent pattern set identification, clustering or classification [1] – we are specifically interested in the case of classification. The base case represents a subset of the complete set of potential requirements for a streaming classifier, as will be made apparent in the following.

Under the specific context of streaming classifiers, many potential drawbacks remain to algorithms proposed to date [1]. For example, previous algorithms are often limited to binary tasks or require in the order of millions of training instances to build a single classifier. Moreover, the accuracy of the stream classifier is often lower than that of the batch equivalent. Ultimately, a stream classifier should also ideally address the issue of *concept drift* where this reflects the fact that the underlying process might be non-stationary. Current methods adopt one of two approaches; they either deploy a change detection metric to the original stream data (e.g., [1]), or – particularly in the case of ensemble style classifiers – detect when variance appears in classifier behaviour (e.g., [18]). In this research our specific interest lies in characterizing how well we are able to address the following **base case**: assuming that labelled training data exists, does the one pass assumption and the properties of any archiving strategy impact negatively on the resulting classifier performance? This is an important question as it implies that not only is fitness evaluation only performed relative to a small subset of training instances at each generation, but there is no recourse for revisiting a training instance unless it is explicitly retained in a finite size archive. Assuming that the base case can be solved, then introducing support for concept drift can be provided by adopting current models for change detection e.g., [1], [18]. This would then scale any proposed algorithm for the base case to the case of streams for which labelling carries a significant cost i.e., only under the case of a change being detected (in classifier or stream behaviour) do you request data labels.

3 Methodology

3.1 Streaming data interface

The streaming data assumption places constraints on how training data can be accessed. Specifically, we assume a sliding window representation (Section 1). From the perspective of a population of GP classifiers under evolution this means that new training instances become available *incrementally* at each generation. A point population, P, is used to define the specific sample of training instances over which fitness evaluation is performed at generation, t. Likewise, at each generation some subset of the point population is replaced by training instances currently available from the stream's sliding window, S^t. A breeder style replacement policy will be assumed, wherein P_{gap} is the number of points replaced. Such an architecture is summarized by Figure 1.

The sampling of training instances from the content of the current sliding window will assume the following process:

1. Define the class label to be sampled with uniform probability (thus we assume that we know how many classes exist);

2. Sample uniformly from the training instances that match that class (again as limited to the content of the sliding window, S^t).

Such a scheme reflects the fact that fitness evaluation at any training epoch is performed relative to the content of the point population, P, *not* the sliding window, S^t. The above two step process is repeated P_{gap} times at each generation, t, thus sampling the content of the sliding window. In addition, we assume that the initial interval over which the point population samples corresponds to 10% of the data from the stream. This corresponds to waiting for a *fixed period of time* to obtain content in the window i.e., streams with a higher bandwidth would have a larger initial sample. The above two step process is used to seed the initial point population. After $\frac{t_{max}}{10}$ generations the S^t content is manipulated by a window protocol where several are discussed in Section 4.1. This follows recent streaming benchmarking practice e.g., [1].

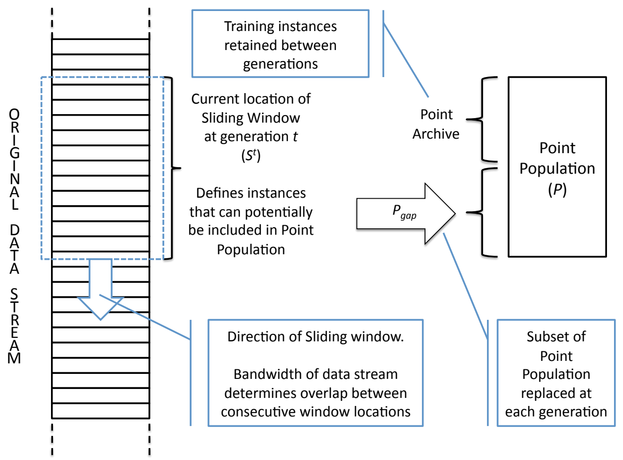

Training instances retained between generations
Current location of Sliding Window at generation t (S^t)
Defines instances that can potentially be included in Point Population
Point Archive
Point Population (P)
P_{gap}
ORIGINAL DATA STREAM
Direction of Sliding window.
Bandwidth of data stream determines overlap between consecutive window locations
Subset of Point Population replaced at each generation

Figure 1: Relationship between streaming data, τ, sliding window, S^t, and point population, P. Only the content of the point population is used for fitness evaluation. A total of P_{gap} records are replaced from the point population at each generation. Pareto archiving defines up to $|P| - P_{gap}$ records retained between generations.

3.2 Pareto coevolutionary archiving

The interaction between GP individuals and point population will be modelled under a Pareto coevolutionary mechanism (e.g., [6, 14, 3]). Possible alternatives might assume sampling under probabilistic frameworks [7], error variance sampling [8] or host-parasite interactions [2]. In the following we summarize the process adopted by the Symbiotic bid-based (SBB) framework for cooperatively evolving GP teams [11, 5]. A two stage process is therefore assumed consisting of Pareto dominance ranking and fitness sharing,[1] a decision in part made on the availability of SBB source code.[2]

Pareto dominance ranking: assumes a Pareto coevolutionary multi-objective optimization approach to archive construction. Members of the point population, p_k denote 'objectives' used to distinguish between the capability of different GP learners under a pairwise test for Pareto dominance, or

$$\begin{array}{c}\forall p_k \in P : G(l_i, p_k) \geq G(l_j, p_k) \\ \text{AND } \exists p_k \in P : G(l_i, p_k) > G(l_j, p_k)\end{array} \tag{1}$$

where P is the point population; l_i and l_j are two individuals (cf., learners) from the GP population currently under evaluation, and; $G(\cdot, \cdot)$ is the task specific reward function returning 0 on an incorrect classification and 1 on a correct classification (see Eqn (4) later).

Thus, Eqn (1) identifies learner l_i as dominating learner l_j *iff* it performs as well on every point and better on at least one point. For a total of $|L|$ learners there are a total of $|L|^2 - |L|$ learner comparisons [3].[3] Such a list of comparisons implies that for point p_k a comparison between learner l_i and l_j has the form of a distinction vector:

$$d_k[L \cdot i + j] = \begin{cases} 1 & if G(l_i, p_k) > G(l_j, p_k) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

If Eqn (2) returns a value of 1 then a *distinction* is said to have been made [6]. Points are penalized for defeating all learners or when it is defeated by all learners i.e., no distinctions result. The fittest learners are naturally those promoted by the Pareto dominance expression of Eqn (1). A basic drawback of the Pareto approach is that as the number of objectives increases, then comparatively weak overlapping behaviours can legally enter the Pareto archive [6, 14, 12]. To this end, SBB adopts the following fitness sharing heuristic with the motivation of maintaining diversity in the points learners solve.

Fitness sharing: The reward function $G(l_i, p_k)$ establishes a vector of distinctions (Eqn. (2)) for each point. The point population can now be divided into two sets: those in the non-dominated front (the archive), $\mathcal{F}(P^t)$, versus those that are not, $\mathcal{D}(P^t)$. Naturally, the decision to limit the number of archives to two is motivated by a desire to balance the computational cost of archive construction against maintaining monotonic progress to an 'ideal'

[1]For a tutorial on Pareto archiving as applied to GP see [9].
[2]http://www.cs.dal.ca/˜mheywood/Code/SBB
[3]This is distinct from the number of fitness evaluations per generation, with fitness evaluation being a more costly process than establishing the comparison vector.

training trajectory [3]. A recent comparison between host-parasite and Pareto archiving indicated that enforcing archive limits can have a significant computational advantage without appearing to impact solution quality [2].

At each generation up to P_{gap} points are replaced from the current stream window using the process of Section 3.1. Two basic conditions hold for targeting point replacement [11, 5]:

- If $|\mathcal{F}(P^t)| \leq |P| - P_{gap}$ THEN: stochastically select points for replacement from $\mathcal{D}(P^t)$ alone.
- If $|\mathcal{F}(P^t)| > |P| - P_{gap}$ THEN: all the dominated points are replaced, plus some subset $P_{gap} - |\mathcal{D}(P^t)|$ of the Pareto archive. To this end, a fitness sharing heuristic is assumed for weighting members of the Pareto archive such that the more unique the distinction the greater the weight, or relative to distinction vector, d_k, of point p_k:

$$\sum_i \frac{d_k[i]}{1 + N_i} \tag{3}$$

where i indexes all distinction vector entries (Eqn (2)), and N_i counts the number of points in $\mathcal{F}(P^t)$ that make the same distinction.

3.3 Symbiotic bid-based GP

The Symbiotic bid-based framework for GP as applied to classification – hereafter SBB – utilizes two populations linked through symbiosis [11, 5]. Specifically, a program (symbiont) population assumes a bid-based GP representation [10] in which each individual consists of a tuple $\langle c, b \rangle$. Thus, c declares a scalar class label selected from the set of labels associated with the task domain ($c \in C$) and b is the associated program that establishes the context for deploying its class label. Conversely, a teaming (host) population identifies combinations of symbionts which will attempt to coexist in a cooperative coevolutionary relationship. A variable length representation is assumed, wherein the number of symbionts necessary to provide an effective cooperative relationship are adapted as part of the evolutionary cycle. Fitness evaluation is only ever performed at the 'level' of the hosts; thus, hosts represent the 'learner' in the above discussion of Pareto dominance (Section 3.2).

Evaluation of a host, $l_i \in L^t$, has the following form and is repeated for each training instance, p_k, as identified by the point population at generation t, $p_k \in P^t$. For all symbionts a member of the host, $s_j \in l_i$, execute their programs w.r.t. point p_k. The host identifies the symbiont with maximum output (the winning bid) or $s^* = arg_{s_j \in l_i} max[s_j.b]$. The winning symbiont gains the right to suggest the class label for the current point, or

$$G(l_i, p_k) = \begin{cases} 1 & if \text{ IF } s^*.c = p_k.t \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $p_k.t$ is the target label for point p_k. Naturally, $G(l_i, p_k)$ is the reward function referred to in Section 3.2.

Algorithm 1 summarizes the breeder model of evolution defining the generic stepwise application of SBB to a fixed length data stream.[4] Initialization establishes the initial sliding window content, S^t from the data stream, τ, and then samples from S^t until $P - P_{gap}$ points have been selected (see Section 3.1 for the specifics of this process). Initialization of the host and symbiont population follows the original SBB algorithm [11, 5]. On entering the main loop the remaining P_{gap} and L_{gap} points and teams are initialized. In the case of the point population the process is the same as employed during the initialization step. The additional hosts, however, are established through the application of variation operators. Crossover copies the common symbionts into the child and probabilistically samples from any remaining symbionts. Mutation style variation operators can vary the host content (probabilistically add (p_a) or delete (p_d) links to current symbiont content). Symbionts can also be cloned, with the content varied through appropriate GP mutation operators [11, 5]. Should a symbiont fail to receive any host indexes then it is implicitly considered to be unfit and deleted from the symbiont population.

The 'Evaluate' function (line 11) follows the above process for evaluating $G(l_i, p_k)$. Once all hosts have been evaluated on all points[5] then the location of the sliding window is incremented relative to the data stream S^t (line 14).

Point replacement is performed under the Pareto archiving model presented in Section 3.2 (line 15). Likewise, a fixed number of hosts, L_{gap}, are removed at each generation (line 16). Thus, members of the host population are either non-dominated (Pareto archive), $\mathcal{F}(L^t)$, or dominated, $\mathcal{D}(L^t)$, implying that the same tests for identifying

[4]The case of a continuous data stream would require change detection in order to trigger the point(s) at which training should be undertaken (Section 3.1).

[5]A process that permits the reuse of previous evaluations as only P_{gap} and L_{gap} points and hosts change at each generation.

Algorithm 1 Overview of the SBB training algorithm as applied to streaming data τ. S^t denotes the set of training instances associated with the sliding window at generation t; τ denotes the streaming data; $|P|$ and P^t are the size and content of the point population respectively; $|L|$ and L^t are the size and content of the host population respectively;.

```
1:  procedure TRAIN
2:      $t = 0$                                              ▷ Initialization.
3:      $S^t$ = INITSTREAM($\tau(t)$)
4:      $P^t$ = INITPOINTS($P, S^t$)
5:      $(L^t, L^t)$ = INITTEAMS($M$)
6:      while $t \leq t_{max}$ do
7:          $P^t$ = GENPOINTS($P^t, S^t$)                    ▷ Add 'gap' size points
8:          $(L^t)$ = GENTEAMS($L^t$)                        ▷ ... and hosts
9:          for all $l_i \in L^t$ do
10:             for all $p_k \in P^t$ do
11:                 EVALUATE($l_i, p_k$)                     ▷ Evaluate fitness
12:             end for
13:         end for
14:         $S^{t+1}$ = SHIFTSTREAM($\tau(t+1)$)             ▷ Resample
15:         $P^{t+1}$ = SELPOINTS($P^t$)
16:         $(M^{t+1})$ = SELTEAMS($L^t$)
17:         $t = t + 1$
18:     end while
19:     return BEST($L^t, P^t$)
20: end procedure
```

hosts for replacement relative to the size of the Pareto archive are also employed. Should symbionts *no longer* receive any host indexes then this is taken to imply that they were only associated with the worst L_{gap} hosts and they therefore 'die'. The symbiont population size is therefore free to float.

Finally, the role of function 'Best' (line 19) is to return the best individual for evaluation against the test partition. Under the original non-streaming case of SBB it was legitimate to perform this activity against the entire training partition. Under a streaming context this would not be appropriate. Re-evaluating the final content of the host Pareto archive against the entire streaming data set would be inconsistent with the purpose of deriving a streaming learning algorithm. In this case we therefore evaluate all hosts currently in the Pareto archive against the current content of the point population. If the Pareto archiving strategy is effective then the points remaining in the Point population should be the most effective for prioritizing the strongest single host. In this last case, the average class-wise detection rate is employed as the fitness function:

$$avg.DR = \frac{1}{|C|} \sum_{c \in C} DR_c(l_i) \tag{5}$$

where C is the set of class labels associated with this task and $DR_c(l_i)$ is the detection rate of host l_i relative to class c. Readers are referred to [11, 5] for additional details e.g., instruction set and variation operators.

4 Evaluation

4.1 Methodology

Four protocols are considered for defining the relationship between training data (τ), sliding window (S^t), and point population (P):

1. Pareto archiving without limits on how the training data (τ) is accessed – this establishes the performance baseline for Pareto archiving under a batch access policy ($S^t = \tau$) and finite Point population (P). Training instances can be visited any number of times during evolution. Hereafter this case is denoted **bat**.

2. Pareto archiving under sequential access to training exemplars, but with indexing to any *previously encountered* training instance possible – This scenario is equivalent to a sliding window, S^t, in which the lower

Table 1: Characterization of benchmarking datasets. Attribute counts appear next to the respective dataset names; Values in parenthesis within the table denote test partition instance counts.

Class	Census (41)	Thyroid (21)	Shuttle (9)
1	187,141 (93,576)	93 (73)	34,108 (11,478)
2	12,382 (6,186)	191 (177)	37 (13)
3	–	3,488 (3,178)	132 (39)
4	–	–	6,748 (2,155)
5	–	–	2,458 (809)
6	–	–	6 (4)
7	–	–	11 (2)

bound remains unchanged (references the first training instance of τ) but the upper bound increases with training epoch, or $|\tau| \times \frac{t}{t_{max}}$, where $|\tau|$ is the size of the entire training partition, t is the current generation, and t_{max} is the last generation. Thus at the last generation the entire data stream can be sampled. Hereafter this case is referred to as **agP**.

3. Pareto archiving under sequential access to training exemplars, and no capacity for revisiting previously encountered exemplars unless they were retained in point population – As per Figure 1, the sliding window, S^t, increments its location in proportion to the training epoch, t. Thus, under the stochastic sampling process of Section 3.1 the location of the sliding window is bound by the interval $[|\tau|(1-w), ...|\tau|]\frac{t}{t_{max}}$; where w is the percent capacity of the sliding window (Section 4.3). As established in Section 3.1 a total of P_{gap} training instances are sampled from the sliding window at each generation. Hereafter this case is denoted **w*xx*P** where '*xx*' takes the value for w.

4. No Pareto archiving under sequential access to training exemplars – this establishes the contribution of Pareto archiving. Instead, P_{gap} points are selected for replacement under the stochastic sampling process of Section 3.1. The parameterization of the sliding window follows points 2 and 3. Hereafter these cases are denoted **agN** and **w*xx*N** respectively.

Naturally, τ would unknown in practice, implying that t_{max} is undefined i.e., training is performed on a continuous basis. Moreover, in cases 3 and 4, the seeding constraint of Section 3.1 provides S^t access to the first 10% of the stream for the first $\frac{t_{max}}{10}$ generations. The benchmarking study of Section 4.4 assumes finite data sets; the above constraints are designed to enforce a single pass through the data set for a fixed number of generations. Thus, under a common generation limit, t_{max}, we can investigate the impact of different class distributions, sliding window sizes and stream bandwidths. Post training evaluation metrics will take the form of avg. Detection rate (Eqn (5)) as calculated across the unseen test partition.

4.2 Datasets

Three imbalanced datasets from the UCI repository are employed for benchmarking purposes (Table 1).[6] Naturally, the streaming context also implies that the distribution of classes within the stream can have a significant impact on classifier performance. The pathological case of class imbalance coupled with all instances of each class appearing in the same temporal location would result in very biased models. In keeping with recent machine learning practice for streaming data benchmarking (e.g., [1]) we therefore stochastically reorder the training stream such that classes are distributed throughout the stream in keeping with their underlying frequency of occurrence (Table 1). Depending on the capacity of the sliding window, w, this implies that each class may or may not be represented at each time step.

4.3 Parameterization

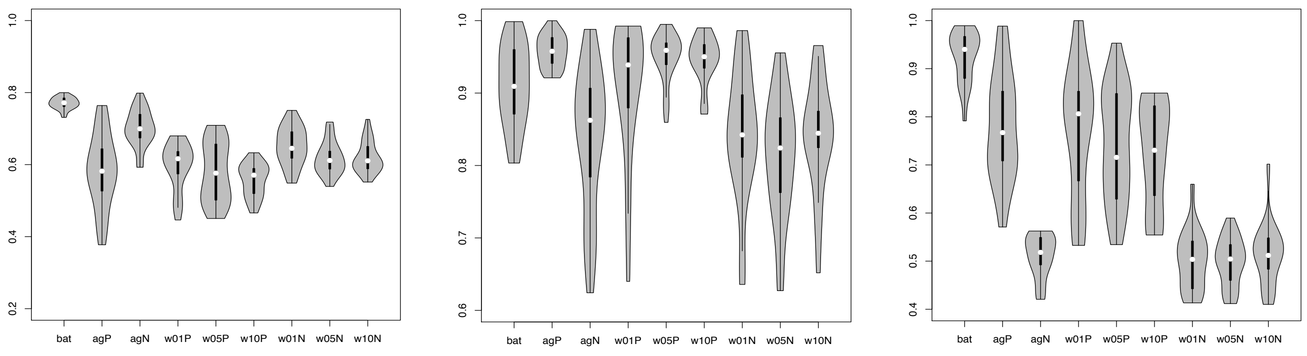
Relative to the three sliding window scenarios identified in Section 4.1, the following parameterizations are assumed:

Maximum number of generations, t_{max}: This defines the number of generations conducted while making a single pass through the data set (Section 4.1), and is held constant across all datasets. The impact of this is that new

[6]http://archive.ics.uci.edu/ml/

Table 2: SBB parameterization.

	Description	Value
$\|P\|$	Point population size.	120
$\|M\|$	Team population size.	120
t_{max}	Number of generations.	30 000
p_d	Probability of learner deletion.	0.7
p_a	Probability of learner addition.	0.7
μ_a	Probability of action mutation.	0.1
ω	Maximum team size.	40
P_{gap}	Point generation gap.	20
L_{gap}	Team generation gap.	60



(a) Census (b) Thyroid (c) Shuttle

Figure 2: Average Detection Rate metric under test data. See section 4.1 for column labels.

training instances become eligible at the rate of ≈ 6.65 per generation under Census; ≈ 1.45 per generation under Shuttle; and ≈ 0.13 per generation under Thyroid (i.e., Total training instance count $\div$ t_{max}).

Window size, w, for the sliding window: Three separate parameterizations are considered: 10, 5 and 1% of the training partition. Naturally, the smaller the window the less opportunity the point population has to sample them before they are shuffled out the sliding window.

SBB parameters: are in some cases increased relative to the original study [11]. In particular, the elitist nature of replacement (w.r.t. point population content) supports higher rates of mutation for adding (p_a) and deleting (p_d) symbionts during reproduction of the host (team) individuals (Table 2). Likewise, the point and host population sizes have also increased, this time on account of the more significant role they play in maintaining a record of training instances and candidate solutions.

In each case a total of 50 runs are performed per experiment.

4.4 Results

In the following we perform a static analysis of the resulting performance post training (Section 4.5), and then investigate the dynamic properties resulting from the streaming context (Section 4.6). Moreover, a recently proposed approach to GP classification in which the *entire training partition* was employed for fitness evaluation at each generation is used to establish an 'upper bound' on what might be expected by way of the avg. DR for each dataset's independent test partition: 80.5%, 95.4% and 98.3% for Census, Thyroid and Shuttle respectively [17].

4.5 Static evaluation

As outlined in Section 4.1 the static analysis considers a total of 5 scenarios: 1) Pareto archiving, but no streaming constraint or the **bat** configuration. 2) Pareto archiving, with streaming constraint, but the opportunity for arbitrary revisiting once encountered, or the **agP** configuration. 3) As per point 2, but with Pareto archiving replaced by uniform selection, or the **agN** configuration. 4) Pareto archiving, with streaming constraint and finite sliding window, or the **w*xx*P** configuration with *xx* denoting window sizes of: 1%, 5%, 10%, hence three distributions. 5) As per point 2, but with Pareto archiving replaced by uniform selection, or the **w*xx*N** configuration from section 4.1 with *xx* denoting window sizes of: 1%, 5%, 10%, hence three distributions;

The binary **Census data set** appears to represent the most difficult scenario (Figure 2(a)), a factor that is also reflected in the case of fitness evaluation performed across the entire training partition [17]; or an avg. DR of 80%. The non-Pareto archiving cases (columns 3, 7–9) return better results than the corresponding Pareto archived cases (columns 2, 4–6). Thus, taken pairwise the case of the aggregated stream window (agN) drops by approx. 10% whereas the finite window cases (w*xx*N) drop by approx. 5%. Pareto archiving under batch access to the training partition (column 1) appears to be largely unaffected. Hence, the relatively poor showing of Pareto archiving under Census is most definitely an artifact of the streaming context. We will return to the source for this performance reduction in the review of the dynamic properties of assuming a streaming context in Section 4.6.

The three class **Thyroid data set** is the smallest data set considered here (Figure 2(b)). This implies that given the fixed generation limit (common to all data sets) then evolution has more time to sample the content of from any particular streaming window location, S^t i.e., S^t content is replace at a slower rate. Pareto archiving (columns 2, 4–6) resulted in significant improvements w.r.t. the corresponding non-Pareto cases (columns 3, 7–9) in all but one parameterization. Moreover, the performance of Pareto archiving with streaming constraints enforced typically performs better than with batch access rights (column 1) on this data set i.e., even under a 1% window size (column 4) the quartile performance is at least as good as the batch case.

Performance on the seven class **Shuttle data set** is summarized by Figure 2(c). A very clear separation again appears between non-Pareto archived (columns 3, 7–9) and Pareto archived (columns 2, 4–6) performance. Thus, significant differences appear between aggregate stream access (column 2 versus 3) and each sliding window variant (columns 4, 5, 6 versus 7, 8, 9). Moreover, the best stream access case corresponds to that of the most restrictive case, or a sliding window of 1%. However, no streaming access scenario is able to reliably approach the batch Pareto case. Moreover, the median performance of all the non-Pareto results indicate that the equivalent of 3 to classes are not being labelled at all. Conversely, median performance of the Pareto enabled models return Detection rates between 70 to 80%, suggesting that a minimum of 5 to 6 classes are detected (1 or 2 classes are missed).

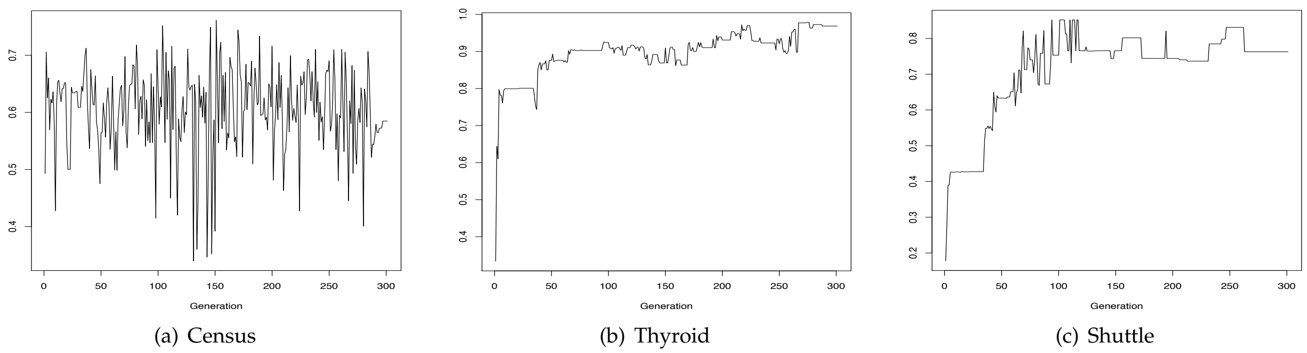
Finally, we note that independent of the data set, there is a strong correlation between performance under the arbitrary revisiting stream constraint (ag*x*)versus strict sliding window constraint for S^t. Thus comparing agP to w*xx*P or agN to w*xx*N under a Wilcoxon non-parametric hypothesis test to each pair of stream configurations (agP versus agN; w*xx*P versus w*xx*N) implies that the non-Pareto case only proves a statistically significant improvement over Pareto archiving for the specific case of agP–agN and w10P–w10N under Census; in all other cases Pareto archiving results in significantly better avg. DR (p-values < 0.005).

4.6 Dynamic evaluation

The preceding section concentrated on evaluating performance relative to the independent test partition post training, i.e. once the entire stream had passed. In doing so it was possible to assess the impact of increasing constraints placed on the form of sampling performed on the training partition both with and without Pareto archiving. From a streaming perspective a clear preference is evident for Pareto archiving under the Thyroid and Shuttle data sets, whereas Pareto archiving appears to negatively impact performance under Census. In this section we investigate how performance on the independent test partition varies during training. This is particularly significant under the sliding window context. Hence, for clarity, in the following we concentrate on the specific case of w05P; where the trends reported are common across w*xx*P, but space precludes explicit documentation.

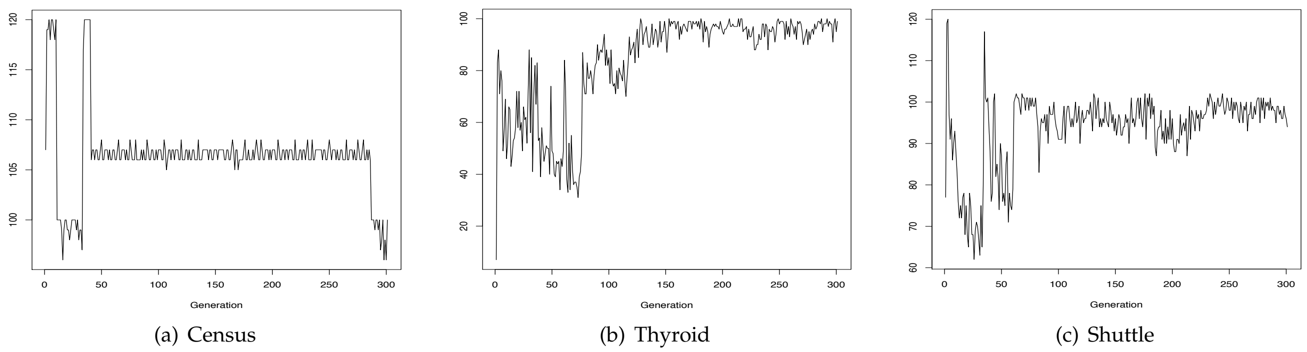
Figure 3 details how performance varies (w.r.t. a specific training run) on the test partition as SBB encounters the training data under the sliding window scenario of w05P. Test data is never used for any aspect of evolution, only as an independent assessment of the degree of regression or improvement at that particular training epoch. Given the computational overhead, a test evaluation is performed once every 100 generations. The contrast between training under Census versus Thyroid and Shuttle is again readily apparent. Specifically, Thyroid and Shuttle continue to improve after the initial period of training against 10% of the data (corresponding to the 0 to 3000 interval on the horizontal axis). Thus, both Thyroid and Shuttle have a distinct step-wise profile in which performance is essentially improving.

Figure 4 summarizes the corresponding utilization of the Pareto archive in the point population, P. Recall that P_{gap} points are replaced at each generation. Thus as long as the size of the Pareto archive remains smaller



(a) Census (b) Thyroid (c) Shuttle

Figure 3: Average Detection Rate of test partition *during* training on the stream using Pareto archiving. Sliding window of 5% in all cases (w05P). For clarity generation axis is $\times 100$.



(a) Census (b) Thyroid (c) Shuttle

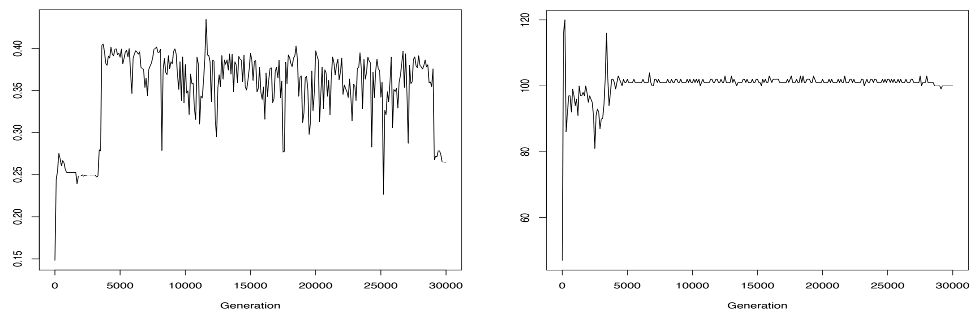
Figure 4: Size of Pareto archive *during* training on the stream using Pareto archiving. Sliding window of 5% in all cases (w05P). For clarity generation axis is $\times 100$.

than $|P| - P_{gap}$ the likelihood of losing potentially good classifiers will be minimized. In the parameterization summarized by Table 2 the Pareto archive should not exceed 100 individuals for this condition to hold. It is apparent that for both Thyroid and Shuttle this condition holds, but in the case of Census the Pareto archive actually matches the size of the Point population before the process of replacing a fixed number of points per generation (P_{gap}) forces the Pareto archive membership to 'sit' at about 105 training exemplars. Doubling the size of the point population had no measurable effect.

Given a 20% error rate, even when employing all the exemplars at each generation for fitness evaluation [17], and a training partition of nearly 200,000 exemplars then a worst case archive in the order of thousands of points might be anticipated. Moreover, we also question what such a rate of error implies. Specifically, this appears to imply that typically 20% of the data is mislabeled or, put another way, the attributes for 20% of the data are not sufficient for distinguishing between the two classes. In order to investigate this hypothesis we introduce a process of re-labelling the Shuttle training data sets with 5, 10 and 20% probability. Thus, a data instance selected with uniform probability is relabelled to that of one of the other classes (also with uniform probability). Figure 5 illustrates how the avg. DR of the test partition and Point archive varies through training for a 5% mislabel rate under the Shuttle data set. It is clear that at best the equivalent of just under 2 classes are learnt during the initial 10% training period (generation < 3000; avg. DR $\approx \frac{2}{7}$), and at best the equivalent of one more class appears thereafter (generation > 5000; avg. DR $\approx \frac{3}{7}$). For completeness, Figure 6 provides a comparison of the post training performance on the test partition (over 20 runs) for cases of 5, 10 and 20% mislabelling. In short, the noise free median performance (column 5, Figure 2(c)) has dropped from an avg. DR of 70% to just above 30% (5% mislabel frequency) and thereafter decaying as the frequency of mislabeling increases.

5 Conclusion

An initial study is conducted to assess the potential contribution of Pareto archiving under GP for a base case in which a regular classification task is limited to a single pass through the training partition under constraints consistent with streaming data. A framework is considered in which a finite archive is enforced, under Pareto and



(a) Test during stream (b) Point archive during stream

Figure 5: Dynamic properties of Shuttle data set with 5% noise under w05P sliding window.

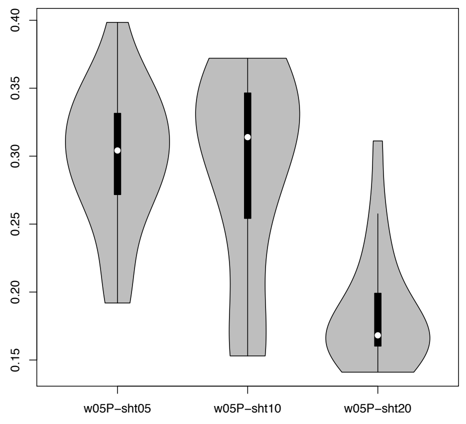


Figure 6: Average Detection Rate of Shuttle data set on test partition.

stochastic policies for archive maintenance. Such an archive is able to retain content independent from that of the sliding window. Three data sets are benchmarked under a common parameterization. The data sets are selected to be representative of class imbalance over multiple classes as well as being of different exemplar and attribute counts.

A study is made of the impact of class mislabeling, a property that all real world data sets will posses to some degree. Significant sensitivity appears to exist, with label error immediately resulting in a larger Pareto archive requirement, or at least when avg. DR represents the performance measure and the class distributions are imbalanced.[7] This in itself might provide the basis for dynamically sizing the Point population to avoid introducing replacement errors and therefore introducing coevolutionary pathologies into the GP population [6, 14, 3, 2]. Moreover, we note that at least two sources of noise can appear: mislabel error (as investigated here), and attribute or sensor error. Future work will continue to investigate both.

Acknowledgements

The authors gratefully acknowledge support from NSERC scholarship and grant programs (Canada).

References

[1] H. Abdulsalam, D. B. Skillicorn, and P. Martin. Classification using streaming random forests. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):22–36, 2011.

[2] J. Cartlidge and D. Ait-Boudaoud. Autonomous virulence adaptation improves coevolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):215–229, 2011.

[7]For example, in the case of the Shuttle data set, the experiment with 5% mislabel frequency resulted in a 30% avg. DR which achieves an accuracy of 75% in this case.

[3] E. D. de Jong. A monolithic archive for pareto-coevolution. *Evolutionary Computation*, 15(1):61–93, 2007.

[4] I. Dempsey, M. O'Neill, and A. Brabazon. Adaptive trading with grammatical evolution. In *IEEE CEC*, pages 2587–2592, 2006.

[5] J. A. Doucette, A. R. McIntyre, P. Lichodzijewski, and M. I. Heywood. Symbolic coevolutionary genetic programming: A benchmarking study under large attribute spaces. *Genetic Programming and Evolvable Machines*, 13(1):71–101, 2012.

[6] S. G. Ficici and J. Pollack. Pareto optimality in coevolutionary learning. In *European Conference on Advances in Artificial Life*, pages 316–325, 2001.

[7] C. Gathercole and P. Ross. Dynamic training subset selection for supervised learning in genetic programming. In *PPSN*, pages 312–321, 1994.

[8] M. Kotanchek, G. Smits, and E. Vladislavleva. Exploiting trustable models via pareto GP for targeted data collection. In *Genetic Programming Theory and Practice VI*, pages 145–162. Springer, 2009.

[9] M. Lemczyk. Pareto-cevolutionary genetic programming classifier. Master's thesis, Faculty of Computer Science, Dalhousie University, 2006. http://web.cs.dal.ca/~mheywood/Thesis/MCS.html.

[10] P. Lichodzijewski and M. I. Heywood. Pareto-coevolutionary genetic programming for problem decomposition in multi-class classification. In *ACM GECCO*, pages 464–471, 2007.

[11] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with symbiotic bid-based genetic programming. In *ACM GECCO*, pages 363–370, 2008.

[12] A. R. McIntyre and M. I. Heywood. Cooperative problem decomposition in pareto competitive classifier models of coevolution. In *European Conference on Genetic Programming*, pages 289–300, 2008.

[13] R. W. Morrison. *Designing evolutionary algorithms for dynamic environments*. Springer, 2004.

[14] J. Noble and R. A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In *ACM GECCO*, pages 493–500, 2001.

[15] M. O'Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):339–363, 2010.

[16] N. Wagner, Z. Michalewicz, M. Khouja, and R. R. McGregor. Time series forecasting for dynamic environments: The DyFor genetic program model. *IEEE Transactions on Evolutionary Computation*, 11(4):433–452, 2007.

[17] S. X. Wu and W. Banzhaf. Rethinking multilevel selection in genetic programming. In *ACM GECCO*, pages 1403–1410, 2011.

[18] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from stream data using optimal weight classifier ensemble. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 40(6):1607–1621, 2010.