On the Impact of Class Imbalance in GP Streaming Classification with Label Budgets

Sara Khanchi[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 EuroGP'16 (LNCS 9594) under Springer copyright 2016
http://link.springer.com/chapter/10.1007\%2F978-3-319-30668-1_3

Abstract

Streaming data scenarios introduce a set of requirements that do not exist under supervised learning paradigms typically employed for classification. Specific examples include, anytime operation, non-stationary processes, and limited label budgets. From the perspective of class imbalance, this implies that it is not even possible to guarantee that all classes are present in the samples of data used to construct a model. Moreover, when decisions are made regarding what subset of data to sample, no label information is available. Only after sampling is label information provided. This represents a more challenging task than encountered under non-streaming (offline) scenarios because the training partition contains label information. In this work, we investigate the utility of different protocols for sampling from the stream under the above constraints. Adopting a uniform sampling protocol was previously shown to be reasonably effective under both evolutionary and non-evolutionary streaming classifiers. In this work, we introduce a scheme for using the current 'champion' classifier to bias the sampling of training instances *during* the course of the stream. The resulting streaming framework for genetic programming is more effective at sampling minor classes and therefore reacting to changes in the underlying process responsible for generating the data stream.

1 Introduction

The streaming data classification task under label budgets introduces a number of constraints that do not appear under the typical offline supervised learning context [9, 5, 14, 1, 7]. Specifically, a streaming data context implies that there is no beginning or end to the data, thus there is no prior partition of the data into training and test sets. Instead, it is necessary to construct a classifier while assuming a limited label budget, i.e. it is too expensive to label all the data, so part of the task of the learning algorithm is to decide which exemplars to request labels for (without exceeding some prior label budget). Naturally, only exemplars for which the streaming data classification algorithm actually requests labels are used for parameterizing candidate classifiers. The remaining data are 'unlabelled' and it is this subset for which a classifier needs to make predictions. Improving the quality of classifier(s) and labelling unlabelled data are therefore not tasks associated with independent prior partitions of the data. Moreover, the data itself might well be generated by a non-stationary process [9, 5]. Thus, the underlying process responsible for creating the data might be subject to sudden shifts or gradual drifts, implying that some form of change detection and / or continuous sampling for labels is necessary in order for classifiers to keep 'up-to-date'.

Naturally, it is not possible to make any guarantees regarding the distribution of class labels within a stream. Moreover, given that streaming classifiers are limited to querying exemplars from a finite 'window' to the stream at any point in time; then, depending on the degree of mixing, the exemplars within the current window location may only be representative of a single class.

In this work, we adopt a generic framework for interfacing genetic programming (GP) to streaming data [9, 17]; hereafter referred as streamGP (Figure 1). The framework identifies four components:

1. the window interface from which new exemplars may be sampled;
2. a sampling policy for deciding which exemplars to request labels for;
3. the data subset against which fitness evaluation is conducted;

4. an archiving policy for deciding which exemplars should be replaced / retained.

In summary, we are interested in considering how decisions that are made regarding the sampling and archiving policies impact on the resulting performance of the classifier. Specifically, we investigate how heuristics for introducing class balance into the data subset can be defined *without* the use of label information.

2 Related work

Streaming data analysis under label budgets represents a topic of growing interest, with several monographs [1, 7] and journal special issues [12] being devoted to the topic. However, until recently there has been little reference to approaches from evolutionary computation that actively construct models (such as GP, learning classifier systems or neural evolution). Conversely, optimization under 'dynamic environments' represents a distinct topic for evolutionary computation with an emphasis on the accurate 'tracking' of multiple 'peaks' in a multi-modal environment, but without requiring generalization to unseen data. As such, there is no requirement to operate under the constraints of a framework for addressing the issue of label budgets. The survey article of [9] reviews developments from the perspective of both non-evolutionary and evolutionary approaches to model building. Particular highlights include:

Ensemble methods provide the ability to incrementally adapt to changes in the stream. Under a GP context adopting an ensemble approach might imply that multiple individuals from the same population coevolve, as in various frameworks for evolving teams of programs [3, 10, 13, 19]. A recent study demonstrated that supporting coevolutionary teaming under GP is particularly appropriate for streaming data contexts [16]. In a sense, modularity enables greater refinement in the credit assignment process, so that rather than having to replace all of a model, only parts of a model require revision. This is particularly important under tasks that lack a 'complete' definition or undergo change. More generally, the capacity to change is related to representations that are in some way 'elastic', with specific authors making the case for the utility of genotypic-to-phenotypic mappings [4] or neutral networks [18] under dynamic environments.

Anytime operation implies that it must be possible to identify at any point in time a 'champion' individual that will attempt to label the stream. At the same time, the development of new individuals may also be taking place, or alternatively, a change detection process is used to trigger the development of new champion individuals (see below).

Diversity maintenance contributes to the ability to react to change in the minimum amount of time. Both non-evolutionary ensemble methods and evolutionary methods appear to benefit from diversity maintenance, but open questions exist around what 'type' of diversity is most appropriate.

Change detection versus sampling represents a requirement unique to the streaming data task under label budgets. Given that models can only be constructed relative to a very limited subset of exemplars at any point in time *and* there is only a limited label budget, then decisions need to be made regarding which data to request labels for. Change detection might be performed relative to stream content in an attempt to detect such changes. However, this will not detect changes that result from the movement in labels from one concept to another. With this in mind, benefits have been associated with adopting combined approaches in which both the periodic uniform sampling for labels is combined with change driven requests for labels (e.g. [14, 20]).

Memory mechanisms are implicit in the use of shared genetic material, support for neutral networks and multi-population models. All forms of memory have a part to play in contributing to solutions to streaming data tasks.

In this work, we will adopt the general framework for applying GP to streaming data from [9] (Figure 1) and make use of the symbiotic bid based (SBB) framework for coevolving GP programs into teams [10]. The capacity of the latter for task decomposition (or constructing modular solutions) has already been demonstrated to be superior to monolithic GP under non-streaming and streaming tasks (see [11] and [16] respectively). Moreover, SBB supports multi-class classification from a single run without having to adopt additional heuristics.

3 Methodology

The advancing stream defines a sequential sequence of exemplars, each of dimension d. Without loss of generality, we assume a non-overlapping window interface SW, each consisting of an equal number of exemplars (Figure 1).

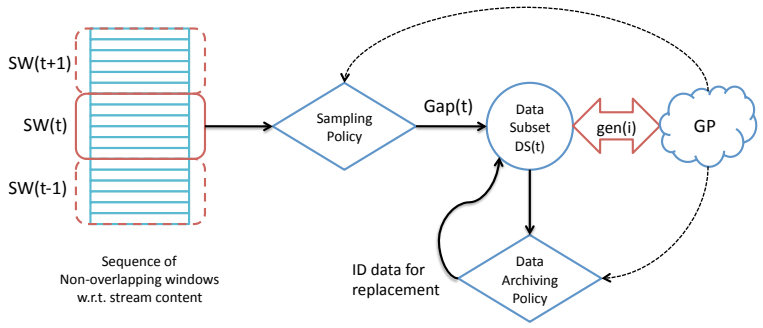


Figure 1: StreamGP framework with proposed additional feedback paths (dotted).

Only the exemplars within the current window position, $SW(t)$, are available for sampling. Label information is *not available* when making the decision regarding which exemplars to sample from $SW(t)$. Moreover, for simplicity we assume that each new window location results in Gap label requests, implying that indexes for window instance and the label request are the same.

Only once the sampling policy identifies $Gap(t)$ exemplars for sampling is label information revealed, i.e. $|Gap| \div |SW|$ denotes the label budget. The data subset, DS, represents the finite sized archive of labeled exemplars used for guiding the training process; thus, $|DS| > |Gap|$. A data archiving policy determines which exemplars are replaced each time a new sample of Gap exemplars are taken from the stream. Once data subset content is defined for the current window location, $DS(t)$, one or more generations of GP are performed. It is relative to the content of the data subset that a champion individual is identified for the purposes of anytime operation (Section 3.1).

Naturally, the sampling policy might be based on a measure designed to detect change between sequential window locations (for a review see [9]). However, this also limits the circumstances under which model reconstruction is initiated. Conversely, assuming a uniform sampling policy (subject to the label budget) has been empirically shown to be difficult to improve on in practice [20, 14]. This was also the approach adopted in the original experiments with the framework of Figure 1 [16, 17].[1]

The question we are interested in explicitly addressing in this work is to what degree the properties of the resulting classifier are biased by decisions made regarding the sampling / archiving policy (Figure 1). The underlying constraint within which such a sampling policy is required to exist is that label information is *not available* when deciding which exemplars to sample. Two **protocols** will be considered:

1. sample with uniform probability up to the label budget, as per earlier studies [16, 17], hereafter the **uniform sampling policy**;
2. make use of the current champion classifier to *suggest labels* and therefore bias the replacement / selection of exemplars, hereafter the **biased sampling policy**.

The first scenario implies that the exemplars within the data subset (against which GP individuals are evolved) will reflect the underlying distribution of exemplars in the stream. The second scenario has the potential to incrementally balance the representation of classes within the data subset. In the following subsections, we develop the framework for anytime operation (champion identification) and then establish the mechanism assumed for reintroducing class-wise sampling of the stream without recourse to any additional label information.

3.1 Anytime operation

As noted in Section 2, streaming data algorithms are required to identify a 'champion' model at any point to label the stream data as it 'passes by' or anytime operation. The only source of information for the purpose of choosing such a champion individual is the current content of the data subset, Figure 1. Thus, once all GP individuals are evaluated against all DS content (or generation, i), a candidate 'champion classifier' can be identified and deployed. Thereafter, a new champion might be identified on concluding each generation. The process operates entirely within the label budget constraint and results in anytime operation. The metric employed for this purpose is that

[1] Earlier work with SBB under streaming data assumed that label information could be used to ensure the data subset was always balanced [15].

of multi-class detection rate, as follows: $DR = \frac{1}{C}\sum DR_i$ where $DR_i = \frac{tp_i}{tp_i+fn_i}$; C is the count of classes present in $DS(t)$; tp_i and fn_i are the counts of true positive and false negative for class i, again relative to exemplars present in $DS(t)$.

Naturally, during the course of a streaming data sequence multiple champion individuals might be identified, but only one champion deployed during any segment of the stream. This is distinct from the style of operation assumed for non-streaming data in which models are constructed from a training partition, a champion individual is identified relative to the entire training partition (or an independent validation partition) and test is performed relative to an independent test partition. None of this is possible under streaming data scenarios because access to the data is very limited (with the process creating the data itself potentially being non-stationary) [9, 5, 1, 7].

3.2 Archiving Policy

The modified archiving policy is designed to target overrepresented classes for replacement by the next sample of Gap exemplars. This means that exemplars representing the minor class(es) are more likely to be retained in $DS(t)$ between consecutive t, whereas exemplars associated with the major classes are prioritized for replacement. Naturally, this also means that the exemplars representing the major classes are more up to date / turn over at a higher frequency.

The modified archiving policy is detailed as follows:

1. Estimate the current class-wise distribution of exemplars within $DS(t)$, or

$$\forall c \in C' : \forall i \in DS(t) \text{ IF } p_i == c \text{ THEN } w_c = w_c + 1 \tag{1}$$

   where, C' are the number of different classes present within $DS(t)$, p_i is exemplar i from the data subset.

2. Normalize class counts (w_c) so distinguishing between under and over represented classes, or

$$\forall c \in C' : w_c = w_c - \frac{|DS| - |Gap|}{C'} \tag{2}$$

3. Mark all cases with $w_c > 0$ so identifying the overrepresented classes and identify the corresponding budget of exemplars for replacement, M_c, or

$$\forall c \in C' : M_c = \begin{cases} w_c \times \frac{|Gap|}{\sum w_c} & \text{IF } w_c > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

4. For each class, mark M_c instances for replacement with uniform probability, subject to a total budget of Gap. Note that in doing so, older instances are replaced first.

5. IF Step 4 marked less than Gap instances from $DS(t)$ for replacement, THEN the remaining instances $Gap - \sum_{c \in C'} M_c$ are identified uniformly across the overrepresented classes until a total of Gap instances are marked for replacement.

3.3 Sampling Policy

Section 3.2 introduced a bias that resulted in the more overrepresented classes being targeted for replacement. Naturally, this has not done anything to increase the chances of sampling instances from the stream corresponding to less frequently sampled classes. The principle constraint is that we have a limited label budget. Our approach will therefore be to make use of the labels supplied by the champion individual, gp^* (identified in support of the anytime operational requirement, Section 3.1) to bias the selection of exemplars for inclusion within Gap relative to the current window location $SW(t)$. Thus, preference will be given to the exemplars that the champion classifier associates with the underrepresented class(es) in the class distribution present in $DT(t-1)$. The resulting sampling policy has the following form:

1. Assume Eq. (1) through (3) from Section 3.2 to identify any underrepresented classes and their associated exemplar counts, w_c. Such a process is performed w.r.t. $DS(t-1)$ content, i.e. after the last updating of the Archive Policy.

2. Use the current champion GP classifier, gp^*, to identify any instances under $SW(t)$ that (potentially) correspond to an under represented class, or

$$\forall p_i \in SW(t) : G(p_i, m) = \begin{cases} m & \text{IF } gp^*(p_i) == M_c(m) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $M_c(m)$ are the subset of classes underrepresented (i.e., those for which $M_c == 0$ from Eq. (3)), $G(p_i, m)$ is a vector of class labels corresponding to the underrepresented instances and m correspond to the class label for an underrepresented class.[2]

3. The non-zero entries of $G(\cdot, \cdot)$ constitute the instances potentially corresponding to the under represented class(es). Sample without replacement until either no instances remain (in $G(\cdot, \cdot)$) or $|Gap|$ instances have been sampled. Such a sampling process is biased to prioritize sampling classes least represented in $DS(t-1)$.

4. If Gap is not yet full, sample from the remaining instances in $SW(t)$ without replacement (uniform p.d.f.).

The above process is enabled *once* some stability is achieved in the identification of champion individuals. For the purposes of the experimental evaluation a generic threshold of the first 2.5% of the generations is assumed across all data sets.[3] During this initialization period the Gap individuals are selected with uniform p.d.f. alone. No such constraint is employed in the case of the Archiving Policy (Section 3.2).

4 Experimental Methodology

An empirical evaluation is performed to investigate the impact of assuming the biased sampling protocol (Section 3) versus a uniform sampling of training instances under fixed label budgets.[4] In each case 20 runs are performed over 4 data sets previously employed for benchmarking streaming data algorithms. Two data sets are artificial data sets with specific non-stationary properties present (i.e., explicitly designed in), whereas the other two data sets represent real-world tasks in which general spatial-temporal properties are assumed to be present. Section 4.1 will summarize the properties of these data sets.

Metrics for performance evaluation under streaming data are in itself the subject of active research [8, 2]. That is to say, streaming data has a dynamic component on account of the model being under continuous development throughout the stream; thus, the performance metric should be able to characterize performance over the *course* of the stream. In this work, we will adopt a count based metric as it is both robust to different class distributions and capable of expressing the dynamic properties of classifier performance during the course of the stream [9, 16]. The specific formulation is presented in Section 4.2. Finally, Section 4.3 establishes a common parameterization for use throughout the study.

4.1 Datasets

A total of four data sets will be assumed in which two are artificially constructed in order for specific non-stationary properties to be embedded within the stream: hereafter Shift and Drift.[5] The **Shift** dataset [20] defines a 5-class task in 6-dimensions in which two decision trees are used to define rules for two *separate* 5-class classification tasks: $C1$ and $C2$. The stream is defined in terms of a sequence of 'blocks'. Each block is composed from β% of exemplars defined by decision tree $C1$ and $(100-\beta)$% of exemplars defined by decision tree $C2$. The first three blocks assume $\beta = 0$% thereafter each block results in β incrementally increasing by 10% until $\beta = 100$%. The **Drift** dataset [6] is defined by a process of gradual variation in which three classes are described by 10-dimensional hyperplanes. Every 1,000 exemplars half of the parameters may undergo variation. Class labels are defined on the basis of whether the hyperplane exceeds a predefined class threshold.

We also make use of the widely used 'electricity utilization' dataset in which the goal is to predict whether the price of electricity (in a region of Australia) are going to increase or decrease relative to a moving average of the last 24 hours. As such this is an example of a real-world task with implicit temporal properties and has received considerable interest from the perspective of the empirical evaluation of streaming algorithms (e.g., [2]). The final dataset is the 'forest cover type' dataset from the UCI repository, but preprocessed to introduce a sequential

[2] Valid class labels appear over the interval $[1, ..., C]$.

[3] Given the later benchmarking parameterization this corresponds to no more than 25 generations.

[4] Previous studies had compared StreamGP under the uniform sampling protocol to non-evolutionary streaming algorithms [17, 16].

[5] Shift and Drift datasets are available from: `http://web.cs.dal.ca/~mheywood/Code/SBB/Stream/StreamData.html`

Table 1: Properties of the benchmarking datasets. D: number of attributes per exemplar, N: cardinality of the stream, k: number of classes present, 'Class distribution' reflects the overall frequency with which each class is represented over the entire stream. No attempt is made to ensure that this ratio is reflected in the window interface used by the classifier to sample stream content.

Dataset	D	N	k	$\approx$ %Class distribution
Shift	6	$6,500,000$	5	$[37, 25, 24, 9, 4]$
Drift	10	$150,000$	3	$[74, 16, 10]$
Electricity	8	$45,312$	2	$[58, 42]$
Cover	54	$581,012$	7	$[49, 36, 6, 4, 3, 1.5, 0.5]$

ordering in the sequence relative to the elevation attribute [14].[6] Table 1 summarizes the generic properties of each data stream.

4.2 Class-wise detection rate

As noted above, given that under streaming data scenarios there are no mechanisms by which stream content can be stratified, then there are no guarantees that window content, $SW(t)$, will even contain exemplars from each class. With this in mind, the following definition for the online estimation of multi-class detection rate is assumed [9, 17]. A per class detection rate is first defined as follows:

$$DR_c(t) = \frac{tp_c(t)}{tp_c(t) + fn_c(t)} \tag{5}$$

where t is the exemplar index, and $tp_c(t)$, $fn_c(t)$ are the respective online counts for true positive and false negative rates, i.e. up to this point in the stream.

The multi-class detection rate now has the form:

$$DR(t) = \frac{1}{C} \sum_{c=[1,\ldots,C]} DR_c(t) \tag{6}$$

Hence, the multi-class detection rate can also be evaluated at any point in the stream.

4.3 Parameters

GP parameterization follows that adopted in previous work (e.g., [16, 17]) and is summarized in Table 2. Moreover, given that for benchmarking purposes the datasets are of a finite length, we enforce label budgets through the use of a fixed number of locations, i_{max}, for the non-overlapping window $(SW(t))$ and knowledge of the dataset cardinality (Table 3). The earlier work also reported that letting GP perform multiple iterations per $DS(t)$ content was beneficial [16, 17]. With this in mind, we perform experiments with a maximum total number of generations of i_{max} and $5 \times i_{max}$.[7] The former implies that one generation is performed per DS update, the latter implies that five generations are performed per DS update; hereafter referred to as **single generation** and **multi-generation** respectively. The instruction set takes the form of:

- Single argument operators: $R[x] = \langle op \rangle R[y]$ where $\langle op \rangle \in \{\cos, \exp, \log\}$
- Two argument arithmetic operators: $R[x] = R[x] \langle op \rangle R[y]$ where $\langle op \rangle \in \{+, -, \div, \times\}$
- Two argument conditional operator: IF $R[x] < R[y]$ THEN $R[x] = -R[x]$

[6] Electricity and Cover Type are available from: `http://moa.cms.waikato.ac.nz/datasets/`

[7] Any more than five resulted in negligible improvement [16].

Table 2: GP Parameters. Mutation rates control the rate of adding / deleting symbionts or changing symbiont action. *DS* and *Gap* refer to the data types in Figure 1. Host population size and gap imply a breeder model of evolution (the worst $Mgap$ hosts are deleted each generation) [10].

Parameter	Value	Parameter	Value
Prob. Symbiont deletion (pd)	0.3	Data Subset size (DS)	120
Prob. Symbiont addition (pa)	0.3	*DS* gap size (Gap)	20
Prob. Action mutation (μ)	0.1	Host pop.	60
Max. symbionts per host (ω)	20	GP gap size $(Mgap)$	20

Table 3: Stream Dataset Parameters. Label Budget is defined as a function of the number of non-overlapping window locations (i_{max}), *DS Gap* size (20) and dataset cardinality (N).

Parameter	# unique *SW* locations (i_{max})	Label Budget
Shift (shift)	1,000	0.3%
Drift (drift)	500	6.7%
Electricity (elec)	500	22.1%
Covertype (cover)	1,000	3.4%

5 Results

Given that the overall detection rate (Eq. 6)) can be decomposed into the contribution from each per class detection rate (Eq. (5)), we can view the independent contributions from each per class detection rate over the course of the stream; hence, providing additional insight into the relative impact of the original uniform sampling protocol versus the proposed biased sampling protocol.

Sections 5.1 and 5.2 review the resulting dynamic multi-class DR as a function of single and multi-generation parameterizations under uniform and biased sampling protocols. Section 5.3 concludes the result section with a static analysis performed in terms of the end-of-stream performance using the overall detection rate (Eq. 6)).

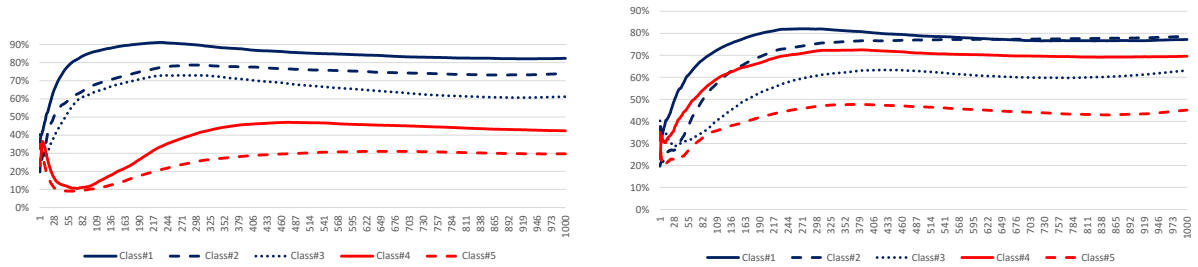
5.1 Single generation performance

Figures 2 and 3 reflect the detection rate of each class over the duration of the stream for the two *artificial datasets* (averaged over the 20 runs). Table 1 details the frequency with which each class is represented. Thus, all figures report class 1 as the most frequently occurring and class *C* as the least frequently occurring. It is readily apparent that the uniform sampling protocol under the Shift dataset explicitly favours the detection of the most frequent classes throughout the stream. Conversely, under the incremental variation of the Drift dataset, the uniform sampling protocol does not reflect this bias, possibly implying that it is more difficult to detect class 2 than 3.

Introducing the biased sampling protocol results in a different preference in class detection rates. Under Shift, the major class (class 1) is still detected most, whereas the second smallest class (class 4) is also detected strongly throughout the stream. Moreover, compared to the uniform protocol, it appears that there is much less difference between the rates at which best and worst classes are detected when using the biased protocol. The Drift dataset resulted in much stronger detection by the biased protocol throughout, albeit with the lest frequent class detected most strongly.

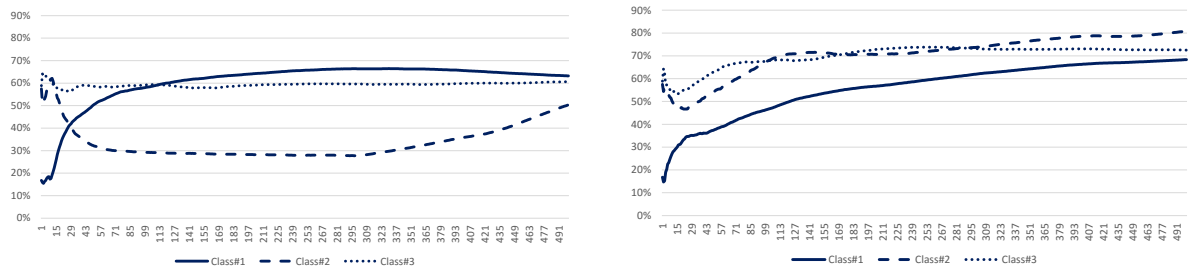
Figures 4 and 5 repeat the dynamic depiction of per class detection rate, this time for the two *real-world* datasets (curves averaged over the 20 runs). Adopting a uniform sampling protocol resulted both classes being detected equally throughout the stream under Electricity (60%). Conversely, the biased sampling protocol initially resulted in a strong symmetry, with a very distinct notch appearing for the duration of the first 2.5% of the stream. This appears to reflect the parameterization choice assumed for delaying the introduction of the Sampling policy (see comment at the end of Section 3.3). That said, the negatively impacted class 2 returns to a detection rate matching that achieved by the uniform framework after $\approx 30\%$ of the stream has passed.

Under the Cover dataset the uniform protocol identified all but two classes with a fixed level of detection rate for the majority of the stream. Class 1 (the major class) is initially detected at a rate of $> 60\%$ before dropping by 10% whereas class 7 is only ever identified right at the end of the stream. Conversely, adopting the biased sampling protocol resulted in class 7 being detected much earlier than under the uniform protocol; likewise for class 6. That said, the two major classes (1 and 2) were always detected more strongly under the uniform framework.



(a) Uniform (b) Biased

Figure 2: Shift dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 1000$, *single generation*. Curves best viewed in colour



(a) Uniform (b) Biased

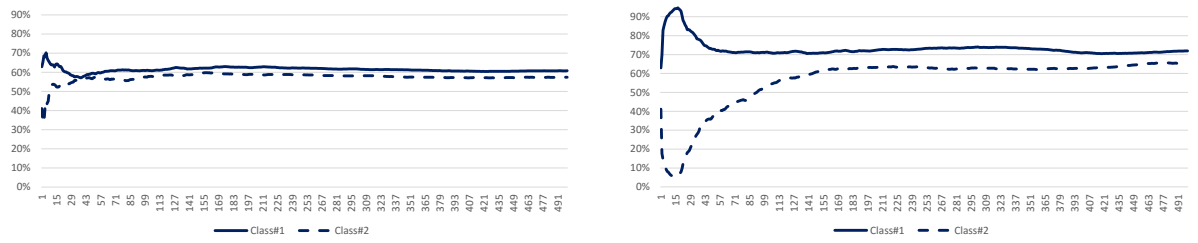
Figure 3: Drift dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 500$, *single generation*

5.2 Multi-generation performance

Adopting a multi-generation parameterization implies that five generations are performed per data subset update ($DS(t)$); thus the label budget is unaffected, but GP is potentially able to react more quickly to change [16, 17]. Other than the addition of multiple generations per $DS(t)$, there are no changes relative to the configuration of the uniform and biased protocols.

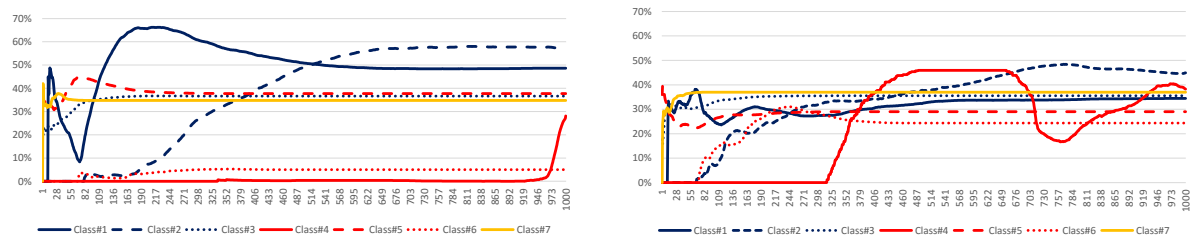
Figure 6 summarizes per class detection rates for the Shift dataset. Relative to the single generation curves (Figure 2) all detection rates are improved, i.e., less variation between the detection of best and worst classes. However, it appears that the biased sampling protocol sees most improvement overall. Under the Drift dataset (Figure 7) all curves are again either improved by the introduction of multi-generation operation or, in the case of the uniform protocol for class 1, negatively impacted. This is interesting, as class 1 is the largest class, thus it might be assumed to see preferential detection by the uniform sampling protocol.

Figure 8 summarizes per class behaviour under the Electricity dataset. Performing multiple generations (per DS update) appears to have very little impact under the uniform sampling protocol, whereas a 5% improvement appears for the detection of each class under the biased protocol. The notch associated with the delayed introduction of the biased Sample policy is again in evidence.



(a) Uniform (b) Biased

Figure 4: Electricity dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 500$, *single generation*



(a) Uniform (b) Biased

Figure 5: Cover type dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 1000$, *single generation*. Curves best viewed in colour

Finally, the Cover type dataset (Figure 9) was also generally improved by the addition of multi-generation operation. Note that the uniform sampling protocol tends to result in a wider spread of per class detection rates, whereas the biased protocol allocated it's resources more evenly across the 7 classes. Also evident is a strong preference under uniform sampling to detect the major class, whereas the biased sampling protocol detects the smallest class the strongest. Naturally, attempting to allocate equal numbers of samples to each class implicitly assumes that all classes are equally difficult to classify. Conversely, in practice the difficulty in detecting a class is not related to the number of instances describing it.

5.3 Overall Detection Rates

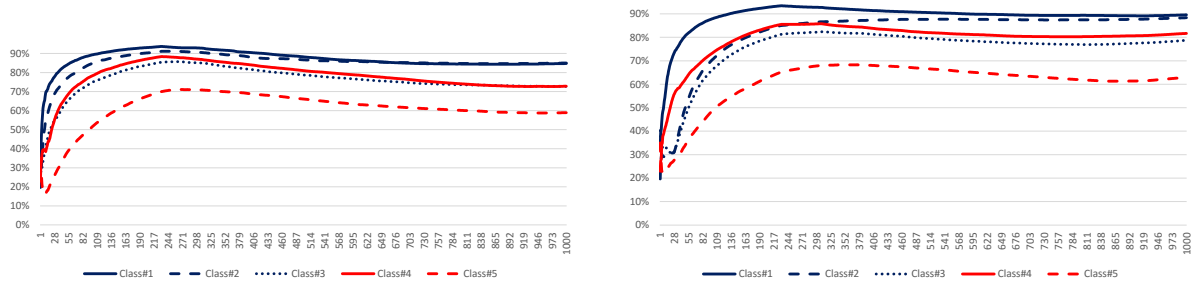
Overall performance of streaming algorithms is generally characterized in terms of the performance metric at the 'conclusion' of the stream (see for example the widespread use of prequential error as measured at the end of the stream [8, 2]). In this case, we can utilize the average class-wise detection rate (Eq. (6)) and then apply a nonparametric Mann-Whitney U test to verify the significance of any difference.[8] Table 4 provides the quantitative summary of this comparison for both the single generation and multi-generation parameterizations under uniform and biased sampling protocols.

In short, under the single generation mode of operation, significant improvements appeared for all but the case of Cover type at the 99% Confidence interval (with the biased sampling protocol preferred). Under the multi-generation mode Cover type was also improved, thus, both algorithms improved with the inclusion of the biased sampling protocol.

6 Conclusion

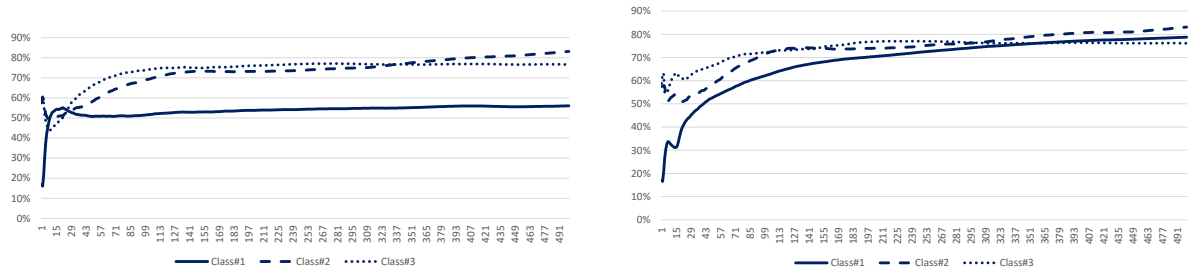
Building classifiers for non-stationary streaming data applications with label budgets represents a new challenge for machine learning in general [9, 5]. Moreover, only a little research has been conducted to this end using genetic programming. In this work, we benchmark a general framework for applying genetic programming to this task.

[8] Violin plots were used to establish that the distributions did not conform to a normal distribution. Space precludes their inclusion.



(a) Uniform (b) Biased

Figure 6: Shift dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 1000$, *multi-generation*. Curves best viewed in colour



(a) Uniform (b) Biased

Figure 7: Drift dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 500$, *multi-generation*

We show that the current champion from the GP population can be used to provide the basis for defining a biased sampling protocol that more rapidly adapts to dynamical properties in the stream, as well as returning stronger classification performance on the under represented classes. This is achieved without requiring additional label information.

Further investigations will be conducted to determine the relative impact of the 'Archiving Policy' and 'Sampling Policy' independently from each other. We also anticipate characterizing at what points there are changes to the champion classifier during the course of a stream and expand the types of data such algorithms are applied to. Moreover, from the application perspective, we have not sort to explicitly address the issue of how delays in applying an 'oracle' to provide labels when requested impact on the quality of the anytime operation of the classifier.

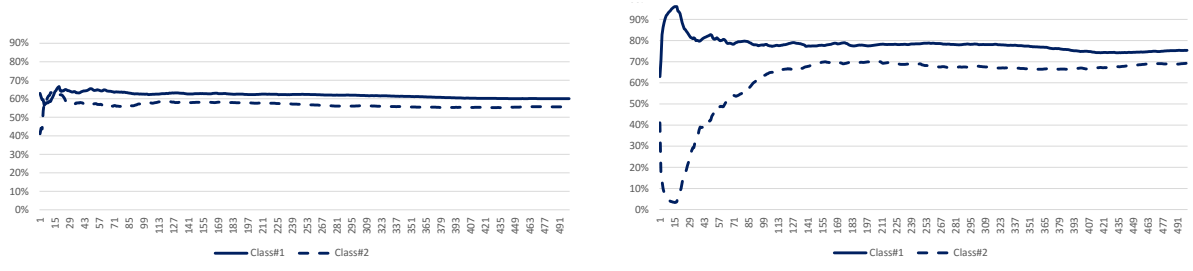
Acknowledgments. This research is supported by the Canadian Safety and Security Program(CSSP) E-Security grant. The CSSP is led by the Defense Research and Development Canada, Centre for Security Science (CSS) on behalf of the Government of Canada and its partners across all levels of government, response and emergency management organizations, nongovernmental agencies, industry and academia.

References

[1] A. Bifet. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*, volume 207 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2010.

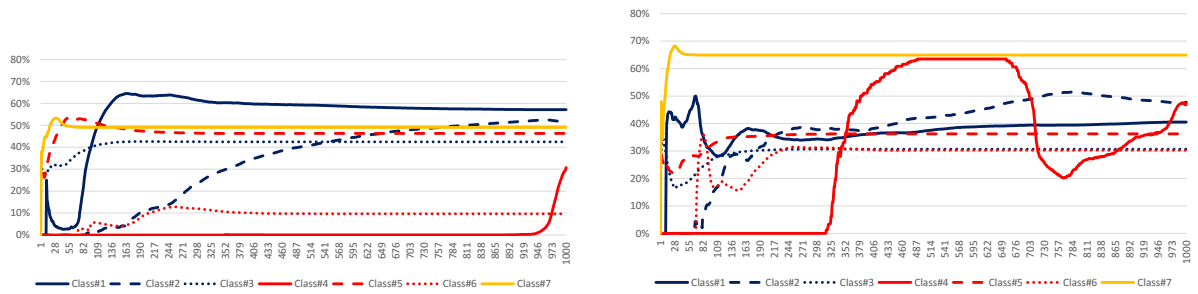
[2] A. Bifet, I. Žliobaitė, B. Pfahringer, and G. Holmes. Pitfalls in benchmarking data stream classification and how to avoid them. In *Machine Learning and Knowledge Discovery in Databases*, volume 8188 of *LNCS*, pages 465–479, 2013.

[3] M. Brameier and W. Banzhaf. Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2(4):381–408, 2001.



(a) Uniform (b) Biased

Figure 8: Electricity dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 500$, *multi-generation*



(a) Uniform (b) Biased

Figure 9: Cover type dataset – Average *per* class detection rate (over 20 runs) under label budget of $i_{max} = 1000$, *multi-generation*. Curves best viewed in colour

[4] I. Dempsey, M. O'Neill, and A. Brabazon. *Foundations in Grammatical Evolution for Dynamic Environments*, volume 194 of *Studies in Computational Intelligence*. Springer, 2009.

[5] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[6] W. Fan, Y. Huang, H. Wang, and P. S. Yu. Active mining of data streams. In *SIAM International Conference on Data Mining*, pages 457–461, 2004.

[7] J. Gama. *Knowledge Discovery from Data Streams*. CRC Press, 2010.

[8] J. Gama, R. Sabastiao, and P. P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90:317–346, 2013.

[9] M. I. Heywood. Evolutionary model building under streaming data for classification tasks: opportunities and challenges. *Genetic Programming and Evolvable Machines*, 16(3):283–326, 2015.

[10] P. Lichodzijewski and M. I. Heywood. Managing team-based problem solving with Symbiotic Bid-based Genetic Programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 363–370, 2008.

[11] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under GP. In *ACM Genetic and Evolutionary Computation Conference*, pages 853–860, 2010.

[12] R. Polikar and C. Alippi. Guest editorial: Learning in nonstationary and evolving environments. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):9–11, 2014.

[13] R. Thomason and T. Soule. Novel ways of improving cooperation and performance in ensemble classifiers. In *ACM Genetic and Evolutionary Computation Conference*, pages 1708–1715, 2007.

[14] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Transactions Neural Networks and Learning Systems*, 25(1):27–54, 2014.

[15] A. Vahdat, A. Atwater, A. R. McIntyre, and M. I. Heywood. On the application of GP to streaming data classification tasks with label budgets. In *ACM GECCO (Companion)*, pages 1287–1294, 2014.

Table 4: End-of-stream Median multi-class detection rates for uniform and biased sampling protocols and corresponding p-value from Mann-Whitney U test

Single generation mode				Multi-generation mode		
Dataset	Uniform	Biased	p-value	Uniform	Biased	p-value
Shift	56.74%	67.5%	1.33×10^{-8}	74.71%	80.37%	1.69×10^{-7}
Drift	58.01%	73.94%	0.0	72.55%	79.48%	0.0
Electricity	59.0%	69.07%	0.0	57.95%	72.6%	0.0
Cover	35.49%	34.21%	0.46	41.9%	42.9%	0.063

[16] A. Vahdat, J. Morgan, A.R. McIntyre, M.I. Heywood, and A.N. Zincir-Heywood. Evolving GP classifiers for streaming data tasks with concept change and label budgets: a benchmarking study. In *Handbook of Genetic Programming Applications*, chapter 18, pages 451–480. Springer, 2015.

[17] A. Vahdat, J. Morgan, A.R. McIntyre, M.I. Heywood, and A.N. Zincir-Heywood. Tapped delay lines for GP streaming data classification with label budgets. In *European Conference on Genetic Programming*, volume 9025 of *LNCS*, pages 126–138. Springer, 2015.

[18] 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.

[19] S. Wu and W. Banzhaf. Rethinking multilevel selection in genetic programming. In *ACM Genetic and Evolutionary Computation Conference*, pages 1403–1410, 2011.

[20] 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.