Properties of a GP Active Learning Framework for Streaming Data with Class Imbalance

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 GECCO'17 under ACM copyright 2017
http://dl.acm.org/citation.cfm?doid=3071178.3071213

Abstract

Active learning algorithms attempt to interactively develop a subset of data from which fitness evaluation is performed. Moreover, the distribution of labeled content within the data subset may adapt over time as genetic programming (GP) individuals improve. The basic goal is therefore to identify the most meaningful subset of data to improve the current model. Under a streaming data context additional challenges exist relative to the non-streaming scenario: non-stationary processes, partial observability, anytime operation. This means that it is not possible to guarantee that the content of the data subset even provides exemplars for each class that could appear in the stream (i.e., different classes appear/disappear at different parts of the stream). With this in mind, an investigation is performed into the impact of adopting different policies for controlling the development of data subset content. To do so, a generic framework is defined in terms of sampling and archiving policies. The resulting evaluation under several large multi-class datasets with class imbalance indicates that adopting random sampling with a biased archiving policy is sufficient for evolving GP classifiers that match or better the current state-of-the-art, particularly when detecting minor classes.

1 Introduction

Streaming data represents a challenging environment for machine learning (ML) algorithms to operate. Data streams appear in many application contexts in which the processes generating the data are non-stationary, e.g. communication networks in which services and protocols change over time, concepts of preference in recommender systems or churn prediction. Moreover, as the data set size increases, it is increasingly likely that the data was collected over an extended period of time (months or years). Thus, the underlying process generating the data is no longer stationary, implying that the underlying assumptions for supervised learning are no longer true [10].

Building (classification) models under streaming data implies that a static partition of training data can no longer be assumed, instead the model adapts interactively with the data stream [18, 7, 6, 9, 13]. This presents multiple challenges for ML in general, a general short list of which might include:[1]

Anytime operation: implies that exemplars from the stream are labeled by the model before the model can adapt to stream content. From the context of genetic programming (GP), this means that a *champion individual* always needs to be on hand to label stream content.

Partial observability: implies that model building progresses incrementally given limited access to stream content, e.g. a sliding or non-overlapping window. This means that the data available does not necessarily characterize all the properties present in the process creating the data. For example, out of a 5 class classification task, the data in the current window location might only describe 1 or 2 classes (and then only partially).

Label budgets: provide the distinction between training and test data. Specifically, unlike non-streaming applications models are constructed incrementally over the duration of the stream. Hence, the exemplars employed for training are sampled over the course of the stream. Any remaining data represents 'test' data. A label budget declares how much of the stream data can be used for training purposes.

Operation under a label budget also means that we are able to function under environments in which labelling all the data is costly. For example, when building models for Botnet detection from network data, the types

[1]Naturally, not all streaming data applications will encounter the same set of constraints.

of attack/communication protocols assumed by Botnets are subject to change at any point in time. Moreover, attackers are explicitly attempting to hide themselves in the network traffic. Constructing detectors offline w.r.t. previously labelled data limits detection to previously encountered attacks/communication protocols. Conversely, requiring the detector to function with the stream under a label budget results in a framework that actively decides what to request labels for, potentially discovering new attacks.

Non-stationarity: implies that the underlying process creating the data is subject to gradual *drift* and/or sudden *shift*. For example, this might reflect variation to a user's interests when visiting social networks or the introduction of new services/applications into network traffic.

Single pass operation: implies that it is not possible to revisit sliding/non-overlapping window locations. Once data moves outside of the current window location (i.e., replaced by more recent content) it cannot be revisited. This is because new data is continuously appearing that replaces older content. Likewise, the champion individual only receives one opportunity to predict the label, $y(t)$ for the current exemplar, $\vec{x}(t)$. True labels are only provided after the prediction, i.e. prequential operation.

Various frameworks have been proposed for enabling GP to address these issues, either separately or as a whole [9]. Particular highlights that inform this work include the use of a *finite sized* data subset (DS) to act as a cache of the most 'informative' exemplars from which GP fitness evaluation is performed and facilitate the identification of a champion individual for anytime operation [20]. Several suggestions have been made regarding label budgets including uniform sampling [23, 19], change detection [8, 16], and active learning [23, 19]. We consider this to represent a generic requirement for some form of sampling policy. Moreover, the issue of non-stationary processes can also be addressed through a combination of the sampling policy (defines which exemplars to add to the DS) and an archiving policy (defines which exemplars to replace from the DS).

An earlier work adopted a specific combination of sampling and archiving policies [12]. In this work, the goal is to isolate the various contributions of each component and identify whether the initial assumptions were correct. In doing so, we are able to explicitly recommend the combination of a uniform sampling policy and biased archiving policy. This appears to be more effective than the previously proposed combination of biased sampling and biased archiving policies while providing a simpler algorithmic approach.

2 Background

In this work we are explicitly interested in constructing models of classification capable of operating under the combination of the four factors highlighted in the introduction: anytime operation, partial observability, label budgets, and non-stationarity. One implication of these constraints are that only some fraction of the stream may be used for constructing a model, i.e. an explicit decision needs to be made regarding what data to query. A sampling policy determines which exemplars need querying. Several schemes have been proposed for this purpose, including: 1) change detection (e.g. the distribution of data between consecutive sliding windows is compared with a threshold of similarity [8] or the distribution of the model output is itself modelled [16]), 2) random sampling, 3) model specific statistics (e.g. how often different leafs of a decision tree suggest a label [21]), 4) active learning [23, 18, 19, 4]. The later case implies that: i) Model building is only performed against the content of the data subset. ii) Predictions from the model can be used to bias the composition of a data subset, e.g. those that introduce most variance [23] or uncertainty [4]. Žliobaitė et al. make the important observation that the impact of drift on the 'strong' predictions is such that sampling should query both the certain and uncertain predictions in order to effectively resist the impact of non-stationarity [19]. Finally, we note that competitive coevolution has also been assessed under streaming data contexts, but with mixed results [2, 1].

The case of active learning is particularly relevant to streaming data as it potentially enables GP fitness evaluation to be decoupled from the raw content of the data stream. Moreover, the underlying distribution of exemplars in the data subset need not be the same as that in the stream. This means that issues such as class imbalance can be directly addressed, i.e. at any point in the stream, it is quite likely that only a subset of classes will be present. A generic framework was recently articulated for addressing the cross section of issues present in streaming data classification under imbalanced data using a combination of non-overlapping window, data subset and biased sampling–archiving policies [12]. However, the contribution of the various components was not explicitly established, nor was it benchmarked against current state-of-the-art.

3 GP Active Learning Framework for Streaming Data

Figure 1 summarizes the overall active learning framework assumed for deploying GP under streaming data applications in which all four streaming data properties from Section 1 are present. A sampling policy (S) determines

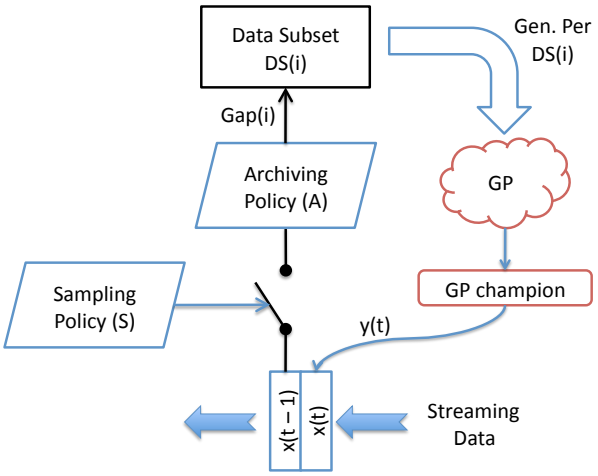


Figure 1: GP active learning framework for streaming data.

Table 1: Streaming GP configurations. Random denotes either sampling or archiving data using a uniform p.d.f. Likewise biased implies either the sampling or archiving data under the corresponding biased policies.

Model	Sampling Policy	Archiving Policy
Random (Rnd)	Uniform	Uniform
Archive	Uniform	Biased
Sample	Biased	Uniform
Both	Biased	Biased

which exemplars from the stream to employ for training purposes. Such a policy has to operate under a label budget. Naturally, exemplars employed for training purposes should not employ label information in determining whether they should be sampled [23, 19]. For example, in the case of a random sampling policy each new exemplar from the stream, $\vec{x}(t-1)$, is sampled with probability $0.0 < \beta \leq 1.0$, where $\beta = 1.0$ implies that all exemplars are employed. Such a scheme would ensure that over the duration of the stream [19]: 1) the label budget is enforced, and 2) the underlying distribution of the data is retained by the samples.

After *Gap* exemplars have been chosen for training purposes they enter the (finite size) data subset (DS). Once the data subset is full, an Archiving policy (A) determines which exemplars are replaced.[2] On updating the data subset with *Gap* exemplars, $\tau(=5)$, GP generations are performed and a *GP champion* individual is identified relative to the current data subset content, $DS(i)$. Once identified, the GP champion provides labels, $y(t)$, for new stream data, $\vec{x}(t)$. Thus, following the initial case of $DS(i=0)$, a champion GP is always available for labeling stream content *before* sampling takes place (or prequential operation). This also implies that the GP champion individual labeling stream content is free to change during the course of the stream.

Table 1 identifies a total of four policy scenarios for GP active learning under the generic framework of Figure 1, summarized as follows:

Random: The next record from the stream, $\vec{x}(t-1)$, is tested for sampling with a uniform p.d.f. (or s). Instances for which $s < \beta$ are sampled. Likewise, after the data subset is full, instances are selected for replacement with uniform p.d.f. Such a scheme enforces the label budget without prior access to label information. Žliobaitė et al. show that the probability density of the (labeled) data subset will approximate that of the (unlabeled) stream as more of the stream is encountered [19].

Archive: As per the Random policy scenario, each record from the stream is tested for sampling with a uniform p.d.f. (or s) and instances for which $s < \beta$ are sampled. However, unlike Random, after the data subset is full, *Gap* exemplars are prioritized for replacement with reference to label information, i.e. once sampled, the label information is available. The 'oldest' exemplars from the over represented class (compared to the current DS distribution of exemplar labels) are replaced by each new labeled exemplar from *Gap*. Thus,

[2]We assume that under the 'cold start' case of an empty DS the first $|DS|$ exemplars are copied straight from the stream to the DS.

following repeated sampling from the stream, the distribution of exemplars in the DS will no longer represent the distribution from the stream. The most frequent classes will see more replacement and each class encountered will tend to be represented equally. The motivation for pursuing such a policy is that with a balanced representation of classes in the data subset there is less likelihood of rewarding 'degenerate' GP classifiers.

Sample: Instead of sampling from the stream with a uniform p.d.f. an additional test is included. This time any record (stochastically) sampled from the stream, $\vec{x}(t-k)$ where $k > 1$, has its corresponding predicted label, $y(t-k)$, provided by the GP champion. We except $\vec{x}(t-k)$ with probability $1/\bar{C}'$ where $\bar{C}' = 1 - C'$ and C' is the proportion of class C' currently within the data subset that matches $y(t-k)$. In short, if the label *predicted* by the champion corresponds to a class under represented in the data subset, it is more likely to be sampled from the stream.[3] Once Gap exemplars have been sampled from the stream, Gap records from the data subset are identified for replacement with a uniform p.d.f. (as per Random).

Both: combines the biased sampling policy from 'Sample' with the biased archiving policy from 'Archive'. This was the *only* scenario that Khanchi et al. [12] considered.

The underlying motivation for attempting to incrementally balance the class-wise distribution of exemplars in the data subset is that, building classification models relative to a balanced class distribution tends to optimize robust performance metrics (such as area under the curve), whereas maintaining the original class distribution leads to optimizing for accuracy, e.g. [22].

3.1 Teaming GP

The teaming GP formulation of symbiotic bid-based (SBB) will be assumed for the generic model of GP [14, 15]. Previous work has demonstrated that such a framework is more effective than monolithic (canonical) GP under off-line classification tasks [15] and streaming classification tasks [20]. In the latter case, the teaming component of SBB is able to more effectively switch useful/ redundant programs in/out of the team, thus react much more effectively to changes in stream content than monolithic formulations of GP.

The SBB framework cooperatively coevolves GP individuals through a bidding mechanism that identifies context for an action, in this case a class label. Each program is assigned a single action at initialization; teams and programs are represented in independent populations. The only constraint on team membership is that there must be at least two programs per team, and there must be two different actions present across all the programs participating within the same team. Thus, teams need not be of the same size, programs can appear in multiple teams, and team complement is entirely a function of evolution.

For example, let team tm_j consist of three programs $\{p_a, p_b, p_c\}$. Fitness of the team is assessed over the exemplars from the data subset, $p_k \in DS(i)$. Given training exemplar p_k all three programs from team tm_j are executed relative to p_k. The *single* program from team tm_j with maximum output on p_k 'wins' the right to suggest its action. The action is a single scalar number that each program is initialized with, and defined set of available class labels $a \in \{1, ..., C\}$. Should the action match that of the actual label for exemplar p_k then a correct classification results, and the fitness function is updated accordingly. SBB supports multi-class operation without any additional features, and has source code available in multiple languages (C++, Java, Python).[4]

3.2 Fitness function and champion identification

Given that the distribution of exemplars in the data subset, $DS(i)$, at any point in time is likely to be imbalanced, a robust multi-class performance metric will be assumed. Note that $DS(i)$ is the only source of data with true label information. The performance metric assumed will take the form of the multi-class detection rate (DR):

$$DR = \frac{1}{C'} \sum DR_j \text{ and } DR_j = \frac{tp_j}{tp_j + fn_j} \tag{1}$$

where C' is the count of classes present in $DS(t)$; tp_j and fn_j are the counts of true positive and false negative for class j, again relative to the class distribution present in $DS(i)$.

Such a definition has no temporal properties and merely reflects the current content of the data subset. Moreover, we note that the champion classifier could potentially change as a function of each variation in the data

[3] In order to ensure that the label budget is strictly enforced, queries are actually made w.r.t. a non-overlapping window ($0 < k \leq L$ where L is the window length), until Gap samples are made.

[4] `https://web.cs.dal.ca/~mheywood/Code/index.html`

Table 2: Generic properties of the streaming data sets. D denotes dimensionality, N cardinality, and k is the number of classes present over the entire duration of the stream

Stream Dataset	D	N	k	$\approx$ Class Distribution (%)
Drift	10	150,000	3	[74, 16, 10]
Shift	6	6,500,000	5	[37, 25, 24, 9, 4]
Cover	54	581,012	7	[49, 36, 6, 4, 3, 1.5, 0.5]

subset content (i.e., as a function of *DS* index i), but never more than this. However, once the first champion classifier is identified, anytime operation is uninterrupted as a champion classifier is thereafter always available.

4 Empirical Methodology

4.1 Comparator Algorithm

The most recent comparator work is that of Žliobaitė et al. in which several policies for streaming active learning are analyzed [19]. The work has been implemented as part of the MoA software suite [3]. Specifically, two policies were identified as particularly effective: Split and Variable uncertainty. The Variable uncertainty method uses classifier confidence (i.e., w.r.t. $y(t)$) to bias the exemplars for which labels are requested using a variable threshold of certainty. In effect, labels are requested for exemplars distributed about the decision boundary between classes. The Split strategy builds two classifiers, one under the variable uncertainty approach the other under the random sampling approach. The model built under the random scheme is used to detect change (thus exemplars added to the data subset conform to the underlying distribution from the stream) and the classifier suggests the label, otherwise the variable uncertainty model suggests the label.

A decision tree represents the model built under this setting. Specifically, exemplars from the data subset (content controlled by either the Split or Variable uncertainty active learning scheme) define the sample for choosing decision tree split attributes. The number of attributes employed for this purpose is actually statistically determined by what is known as the Hoeffding bound [10]. The resulting Hoeffding decision tree (DT) is widely employed in streaming data applications [7], and performed well with the active learning schemes of Žliobaitė et al. [19].

4.2 Data sets

Three data sets will be employed, two artificially designed to explicitly embed known non-stationary properties into the stream: Shift and Drift. The third represents the Forest Cover type data set (hereafter 'Cover'), albeit ordered using the elevation attribute as per the study of Žliobaitė et al. [19]. All three data sets represent multi-class classification tasks with considerable amounts of class imbalance (Table 2).

The Shift dataset models non-stationary properties in the stream by sudden step changes in the distribution between two decision trees (R_A and R_B) used to create the data as per [23]. Specifically, each decision tree defines a stochastic process for creating a five class classification problem. The first 300,000 records of the stream sample R_A. Thereafter, each consecutive 100,000 records is constructed using a combination of R_A and R_B or IF $\gamma <$ rnd THEN R_B ELSE R_A. The threshold $\gamma = 0.1$ in the first set of 100,000 records, and thereafter increases by 0.1 at each consecutive set. At the last set only data constructed using R_B is present.

The Drift dataset is created using a set of d-dimensional hyperplanes which incrementally change position every 1,000 records. At each change event half of the parameters undergo variation (the set of parameters is also chosen stochastically). Labels are defined relative to a (normalized) threshold declared in terms of the hyperplane locations. Further details and the parameterizations for creating the data follow from [21].[5]

4.3 Stream GP parameterization

Stream GP parameterization follows that adopted by [12] and summarized in Table 3. Thus, fitness evaluation is performed against 120 exemplars, and 20 exemplars are replaced at a time (*Gap*), or a 'DS update'. Likewise the

[5]The work of [12] also employed Shift and Drift and have made the datasets available publicly `https://web.cs.dal.ca/~mheywood/Data/index.html`.

Table 3: GP Parameters. Mutation rates control the rate of adding/deleting programs or changing an action. DS and Gap refer to the data types in Figure 1. Team population size and gap imply a breeder model of evolution (the worst $Tgap$ teams are deleted each generation) [14].

Parameter	Value
Data Subset size (DS)	120
DS gap (Gap) and GP gap ($Tgap$)	20
Team pop. size (P_{size})	120
Max. programs per team (ω)	20
Prob. Program deletion (pd) or addition (pa)	0.3
Prob. Action mutation (μ)	0.1

team population is 120, of which 20 are replaced per generation ($Tgap$). Khanchi et al. introduced the concept of performing multiple GP fitness evaluations per DS update ($\tau = 5$). The probabilities of deleting/adding a program to a team are relatively high in order to help promote diversity.

4.4 Performance metrics

Given the ease with which rate based metrics may explicitly quantify multi-class performance [11], we will adopt such a metric for this work. Moreover, rate based metrics can be updated incrementally during the course of the stream [9, 17]. The starting point is generally the confusion matrix in which the leading diagonal represents the count of true positives (for each class). Moreover, the remaining distribution of false negative counts w.r.t. each column characterizes how the hypothesized/predicted labels were distributed across the remaining classes. Thus, given the confusion matrix for some point in the stream, t, each class is characterized in terms of the corresponding detection rate at that point in the stream:

$$DR_c(t) = \frac{tp_c(t)}{tp_c(t) + fn_c(t)} \tag{2}$$

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. Note that $fn_c(t)$ reflects the total number of predictions falling across *all other* classes. Moreover, as we are estimating detection rate for each class, the false positive rate of class c is $\approx \sum_i 1 - DR_i(t) \forall i \neq c$.

The multi-class detection rate assumed by this work now has the form:

$$DR(t) = \frac{1}{C^*} \sum_{c=[1,\ldots,C^*]} DR_c(t) \tag{3}$$

where (for continuity) we assume that C^* reflects the true count of total number of classes encountered over the course of the stream.[6] As $0.0 \leq DR_c(t) \leq 1.0$ for all t, we now have a metric for characterizing multi-class performance at any point in the stream. Each class will contribute a maximum of $\frac{1}{C^*}$ to $DR(t)$. Thus, should 90% of the stream represent a single class (the major class), a degenerate classifier might label the entire stream the major class. Under an accuracy or prequential performance metric this would tend to result in an accuracy (prequential error) tending towards 90% (10%), whereas the multi-class detection rate would never exceed $\frac{1}{C^*}$.

5 Results

Experiments will be performed under two label budgets, or $\beta = \{0.05, 0.005\}$, i.e. 5% and 0.5%. This implies that after the cold start,[7] the number of exemplars sampled over the duration of the stream is $\approx \beta N$. Moreover, a prequential model of operation is enforced in which the model (i.e. GP champion) provides the prediction, $y(t)$, for each record. Performance is therefore a distribution of detection rate per class over the entire stream, where the underlying goal is to maximize the $DR(t)$ (Section 4.4). We can now also consider performance from the perspective of the per class detection rate, i.e. the *dynamic* behaviour across the stream. Having established the impact of different design decisions on classification performance, we go on to analyze how the same design

[6] If the true number of classes is unknown, this just means that a stepwise effect appears in the metric each time a previously unseen class is encountered.

[7] The first $|DS|$ records from the stream fill the data subset.

Table 4: Median stream $DR(t = N)$. Bold indicates a best case result for each test condition. Any policy marked with an '*' denotes a policy with a statistically significant worse result (following Friedman test and Nemenyi post-hoc test). R_j denotes the average rank for each policy.

All classes ($\beta = 5\%$)				
Dataset	Shift	Drift	Cover	R_j
Both	94.4	72.1	**41.0**	2.33
Archive	95.1	**72.7**	40.1	**2.0**
Sample	93.8	63.9	37.9	5.0
Rnd*	90.8	66.4	31.8	5.66
Split	**97.8**	68.4	38.5	2.67
Var	96.3	69.8	37.4	3.33
Minor Class ($\beta = 5\%$)				
Both	87.7	72.6	**56.3**	2.67
Archive	93.2	**76.3**	55.2	**2.0**
Sample	87.3	60.2	50.9	4.17
Rnd	80.8	73.6	19.0	4.67
Split	**97.3**	54.1	50.9	3.5
Var	94.5	60.0	44.4	4.0
All classes ($\beta = 0.5\%$)				
Both	84.4	52.0	26.0	4.0
Archive	**85.9**	54.2	26.8	**2.67**
Sample	83.3	47.2	28.3	3.67
Rnd	80.3	51.1	26.8	4.33
Split	78.1	53.3	**28.5**	3.33
Var	79.1	**56.9**	27.0	3.0
Minor Class ($\beta = 0.5\%$)				
Both	68.6	48.7	38.6	2.33
Archive	**78.9**	**59.3**	**40.6**	**1.0**
Sample	68.0	42.2	34.1	3.33
Rnd	65.9	52.5	27.3	4.0
Split*	66.8	29.8	2.9	5.0
Var*	66.2	34.2	0.1	5.33

decisions are reflected in the capacity to influence the distribution of exemplars in the data subset (Section 5.3). Both GP and non-GP runs are collected over 20 independent runs.

5.1 Overall classification performance

$DR(t)$ provides a single scalar value expressing multi-class detection rate over the duration of the stream (Eqn. (3)). Moreover, not all classes have the same cost. It frequently transpires that detection of the least frequent class costs more than detecting the most frequent (hereafter minor and major class respectively). With this in mind, we build $DR(t)$ for overall performance and minor class alone for each data set and both values of β (Table 4).

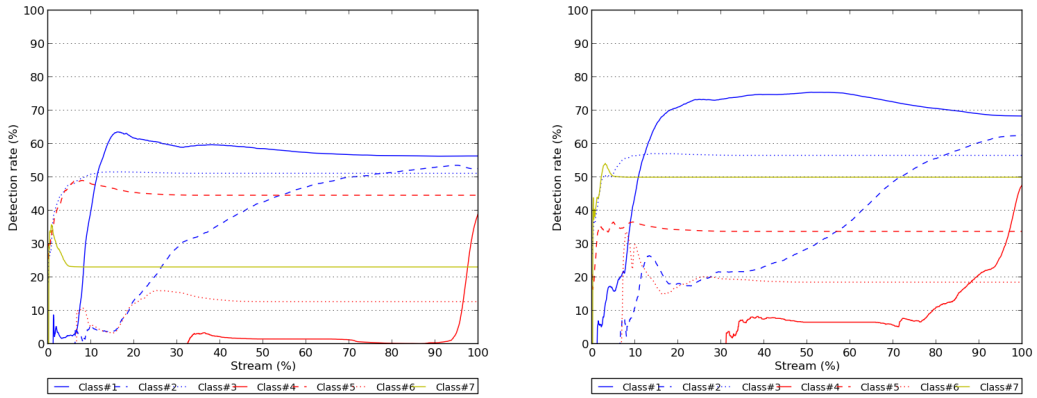
The Friedman test provides a non-parametric equivalent to the repeated-measures ANOVA, and therefore represents a more robust approach for testing multiple algorithms over multiple datasets [5, 11]. Specifically, the Friedman statistic (χ_F^2) first identifies the average rank, R_j, of each algorithm across the set of datasets (lower is better) and measures the degree to which a pattern in the ranks exists or not. In this case there are $k = 6$ algorithms and $n = 3$ datasets. The null-hypothesis is rejected if a pattern exists. Moreover, the Friedman statistic is typically renormalized in the form of the F-distribution (F_F) with $k - 1$ and $(k - 1)(n - 1)$ degrees of freedom [5, 11].

Should the null-hypothesis be rejected, then a Nemenyi post-hoc test defines the critical difference: $CD = q_\alpha \sqrt{k(k+1) \div (6n)}$ by which models should differ for a significant difference to appear. Table 5 summarizes the result of applying the Freidman test under the all class and minor class scenarios. Table 4 indicates the following general trends:

- All algorithms perform well under Shift at the 5% label rate, even for the minor class. However, at the 0.5% label rate both Hoeffding DT policies (Split and Var) undergo the most reduction, especially under the minor

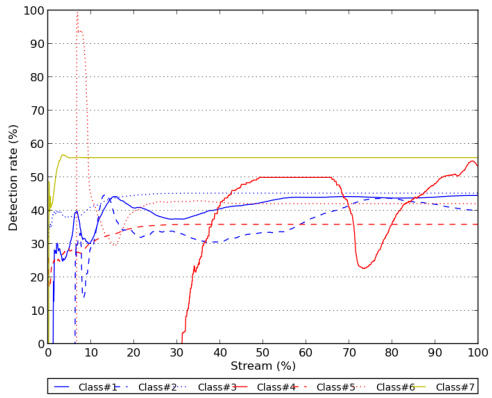
Table 5: Freidman test χ_F^2 and corresponding value for F-distribution F_F. The critical value $F(5,10)$ for $\alpha = 0.1$ is 2.522. Bold font indicates F_F for which the null-hypothesis is rejected.

Percent labels (β)	5% all	5% minor	0.5% all	0.5% minor
χ_F^2	9.67	4.28	1.67	11.57
F_F	**3.63**	0.8	0.25	**6.75**

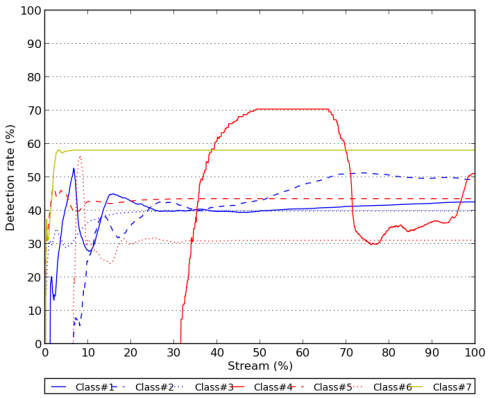


(a) StreamGP–Rnd

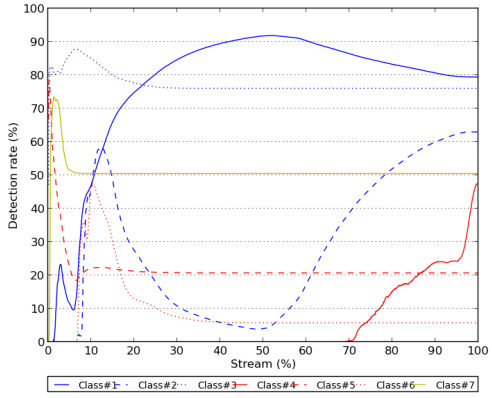
(b) StreamGP–Sample



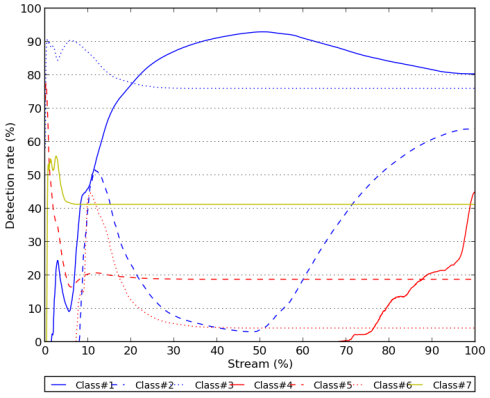
(c) StreamGP–Archive



(d) StreamGP–Both



(e) Hoeffding–Split



(f) Hoeffding–Var

Figure 2: Multi-class Detection Rate of StreamGP and Hoeffding decision tree on Cover dataset for $\beta = 5\%$

class scenario. Similar patterns of behaviour appear for Drift and Cover. The resulting null-hypothesis test is rejected for 0.5% at the minor class scenario (Table 5). A Critical Difference for the Nemenyi post-hoc test identifies Hoeffding DT policies as significantly worse than any Stream GP policy.

- Under the higher (5%) label budget, the control Stream GP policy of random sample and archiving (Rnd) is consistently ranked last, so much so that the null-hypothesis is rejected and the post-hoc test identifies Rnd as significantly worse than the remaining algorithms.
- Generally, the best Stream GP policies are Both and Archive, with Archive always receiving the best ranking irrespective of test scenario. Both was ranked second for all but the 0.5% test over all classes.

In short, the most effective active learning polices for use with GP under streaming data are Archive and Both. These two policies are at least as good as the best (non-GP) comparator algorithms and can be better, particularly in the case of minor class performance.

5.2 Class-wise detection rates

The overall performance evaluation of Section 5.1 hides many properties behind a single scalar number (in this case multi-class detection rate). When the datasets are multi-class and a single performance measure is used to characterize the ability of an algorithm to adapt over the course of a data stream, there are many behaviours that can result in the same scalar measure of performance. Figure 2 summarizes how detection rate for each class adapts over the stream for the four configurations of active learning in Stream GP (5% label budget). Class 1 through 7 are ordered largest (class 1) to smallest (class 7), see Table 2 for the overall distribution.

Figure 2(a) represents the base case in which records are sampled from the stream and replace instances from the data subset stochastically. It is no surprise that the three most frequent classes are detected most effectively, albeit with class 3 detected much more effectively than class 2 throughout the majority of the stream. Class 4 is barely detected at all, whereas class 7 (representing 0.5% of the dataset) is detected at a rate of 20% throughout the stream. This seems to indicate that class 4 is particularly non-stationary whereas class 7 is essentially stationary.

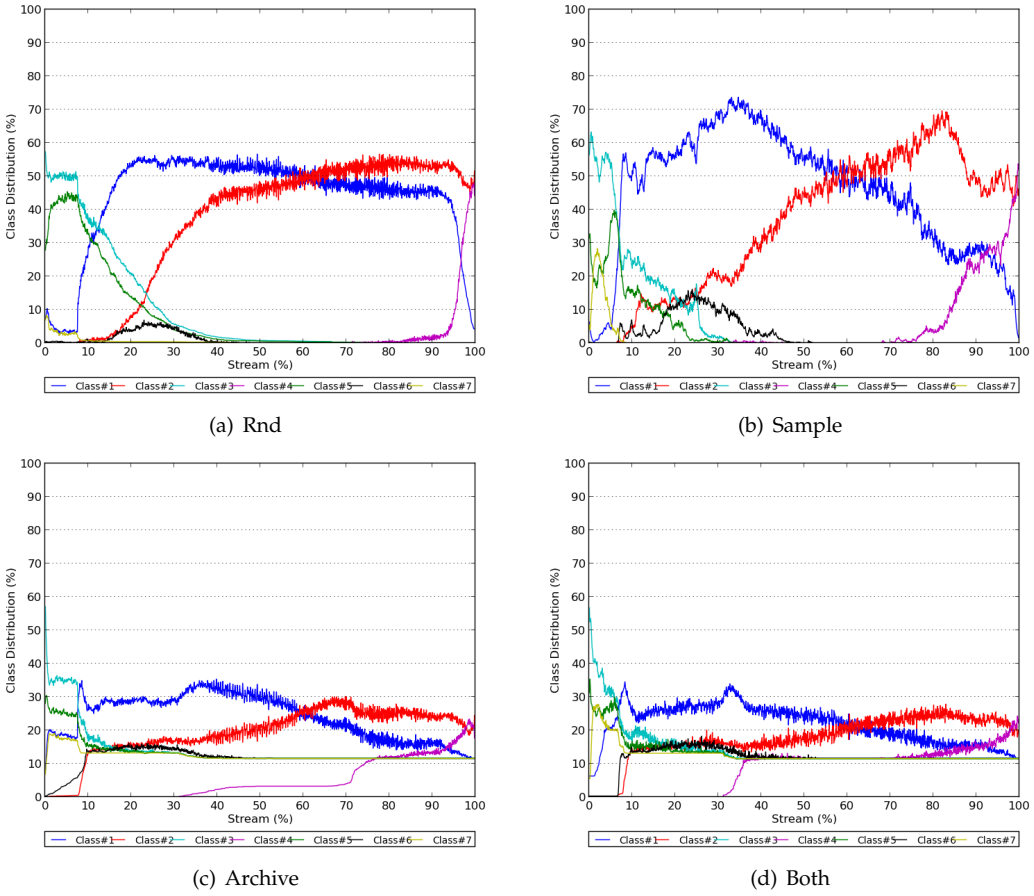
Introducing biased sampling of records (Sample) results in better detection for all classes other than class 5 (Figure 2(b)). Switching back to random sampling from the stream, but introducing targeted replacement of exemplars from the data subset (Archive) results in all the minor classes being detected with a minimum DR of $\approx 35\%$ (as opposed to less than 10% under Rnd or Sample). Supporting both biased sampling and archiving (Both) improved class 4 in particular (Figure 2(d)). However, Class 6 is now detected at a rate of 30% as opposed to 40% under Archive.

Figures 2(e) and 2(f) summarizes the class-wise detection rate for Hoeffding decision tree (DT) under Split and Variable uncertainty active learning policies. Stream GP detection rates for class 1 and 3 are approximately 40% whereas the Hoeffding DT combination returns about 80% detection rates for class 1 and 3. Conversely, Stream GP detected class 2 at a relatively constant rate throughout the stream, whereas Hoeffding DT experienced considerable fluctuation. Moreover, all the remaining classes (4 through 7) were never detected at the rates achieved when Stream GP assumes Archive or Both active learning policies.

5.3 Data subset distributions

Classification performance provides one approach for quantifying the impact of the various Stream GP active learning policies. In this section, we assess the different active learning policies through their impact on the distribution of exemplar classes retained in the data subset *during* the stream. Space precludes analysis over all three data sets, so for consistency with Section 5.2, we concentrate on the Cover dataset (similar observations appearing for Shift and Drift).

Figure 3 details the resulting representation of exemplars from each class during the stream for each active learning policy in Stream GP. Two basic behaviours are now evident. That illustrated by Rnd and Sample (Case 1) versus that by Archive and Both (Case 2). Under Case 1 the two major classes dominate the distribution of exemplars retained within the data subset over the course of the stream. Conversely, the active learning policies of Case 2 are able to achieve a much more balanced distribution of the exemplars retained per class. Two features distinguish Archive from Both. Under Archive, class 4 (which appears for the first time around the 30% mark in the stream) only sees gradual representation over the interval between 40 and 70%. Conversely, the Both active learning policy manages to immediately populate the data subset to slightly less than $\frac{100}{7}\% \approx 14\%$ of the data subset. In effect, the predictions from the GP champion for class 4 were accurate and therefore resulted in new



(a) Rnd (b) Sample

(c) Archive (d) Both

Figure 3: Distribution of exemplar classes in StreamGP data subset on Cover dataset for $\beta = 5\%$

instances of class 4 being quickly prioritized for inserting into the data subset. We associate this property with the 70% DR of class 4 during the mid-point of the stream for SteamGP–Both combination (Figure 2(d)).

There is also a difference between Archive and Both in the way that class 6 is addressed. Class 6 is actually present from the beginning of the stream. However, the biased nature of the Both policy results in this class being ignored until $\approx 8\%$ of the way through the stream. Conversely, the Archive policy samples data for inserting into the data subset with uniform probability. This results in the earlier recognition of this class, ultimately resulting in the SteamGP–Archive combination detecting class 6 at a 10% higher detection rate.

6 Conclusion

Design decisions for GP active learning as applied to streaming data classification tasks is formulated in terms of a sampling policy and an archiving policy. Specifically, fitness evaluation is performed against a data subset, as opposed to the stream directly. The sampling policy defines what exemplars to select for inclusion within a data subset, whereas the archiving policy defines how to replace exemplars from the data subset with new content. Given a base case of a uniform stochastic process for both, we incrementally introduce biased sampling of the stream (using label predictions from the champion classifier) and biased replacement of exemplars from the over represented class(es) in the data subset, resulting in the data subset class distribution incrementally becoming more balanced.

Two active learning configurations result in particularly robust GP streaming classifiers: Both and Archive. Moreover, both policy combinations match or better current state-of-the-art, particularly respect to the detection of minor classes. We also investigate the capability of the framework to actively balance (or not) the distribution of exemplars in the data subset during the course of the stream. Finally, we note that the Archive configuration provides results as good as the more complex Both policy combination, thus facilitating true online operation (Both assumes a non-overlapping window interface). As such this improves on the earlier configuration as proposed by Khanchi et al. [12].

In the case of future work, we are interested in deploying such schemes to the detection of botnet traffic, i.e. a specific application scenario in which the operation under label budgets and the detection of minor classes is extremely important.

7 Acknowledgements

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. Atwater and M. I. Heywood. Benchmarking pareto archiving heuristics in the presence of concept drift: diversity versus age. In *ACM Genetic and Evolutionary Computation Conference*, pages 885–894, 2013.

[2] A. Atwater, M. I. Heywood, and A. N. Zincir-Heywood. GP under streaming data constraints: A case for Pareto archiving? In *ACM Genetic and Evolutionary Computation Conference*, pages 703–710, 2012.

[3] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[4] M.-R. Bouguelia, Y. Belaïd, and A. Belaïd. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters*, 70:38–44, 2016.

[5] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

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

[7] J. Gama. A survey on learning from data streams: Current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55, 2012.

[8] J. Gama, P. Madas, G. Castillo, and P. Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence*, volume 3171 of *LNCS*, pages 286–295, 2004.

[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] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *ACM International Conference on Knowledge and Data Discovery*, pages 97–106, 2001.

[11] N. Japkowicz and M. Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

[12] S. Khanchi, M.I. Heywood, and N. Zincir-Heywood. On the impact of class imbalance in GP streaming classification with label budgets. In *European Conference on Genetic Programming*, volume 9594 of *LNCS*, pages 35–50, 2016.

[13] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[14] 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.

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

[16] P. Lindstrom, B. MacNamee, and S. J. Delany. Drift detection using uncertainty distribution divergence. *Evolving Systems*, 4(1):13–25, 2013.

[17] A. Loginov, M. I. Heywood, and G. Wilson. Benchmarking a coevolutionary streaming classifier under the individual household electric power consumption dataset. In *IEEE-INNS Joint Conference on Neural Networks*, pages 1–8, 2016.

[18] M. Sugiyama and M. Kawanabe. *Machine Learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT Press, 2012.

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

[20] 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.

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

[22] G. M. Weiss and F. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–345, 2003.

[23] 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.