Tapped delay lines for GP streaming data classification with label budgets

Ali Vahdat[1], Jillian Morgan[1], Andrew R. McIntyre[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'15 under Springer copyright 2015
http://link.springer.com/chapter/10.1007\%2F978-3-319-16501-1_11

Abstract

Streaming data classification requires that a model be available for classifying stream content while simultaneously detecting and reacting to changes to the underlying process generating the data. Given that only a fraction of the stream is 'visible' at any point in time (i.e. some form of window interface) then it is difficult to place any guarantee on a classifier encountering a 'well mixed' distribution of classes across the stream. Moreover, streaming data classifiers are also required to operate under a limited label budget (labelling all the data is too expensive). We take these requirements to motivate the use of an active learning strategy for decoupling genetic programming training epochs from stream throughput. The content of a data subset is controlled by a combination of Pareto archiving and stochastic sampling. In addition, a significant benefit is attributed to support for a tapped delay line (TDL) interface to the stream, but this also increases the dimensionality of the task. We demonstrate that the benefits of assuming the TDL can be maintained through the use of oversampling without recourse to additional label information. Benchmarking on 4 dataset demonstrates that the approach is particularly effective when reacting to shifts in the underlying properties of the stream. Moreover, an online formulation for class-wise detection rate is assumed, where this is able to robustly characterize classifier performance throughout the stream.

1 Introduction

Incremental learning from streaming data represents a new challenge for algorithms applied to classification tasks [14, 16, 11, 10]. Such tasks are non-stationary (the underlying process creating the data changes over the course of the stream), have limited capacity for revisiting previously encountered data (single pass constraint), generally present a very imbalanced class distribution (care of the sliding window access to the data) and are subject to a labelling budget (it is prohibitively expensive to label the stream).

In this work we revisit a general architecture previously proposed for applying genetic programming (GP) to streaming data classification tasks under label budgets [19]. The framework assumes a non-overlapping window interface (of length L) to the stream consisting of a continuous sequence of $\vec{x}(t)$ to $\vec{x}(t-L-1)$ exemplars. A subset of instances are stochastically sampled from the current window location and placed into a data subset (DS). Only these exemplars have their label information requested. A training cycle is then performed relative to the current DS content. The authors make use of Pareto archiving to prioritize exemplars within the DS for replacement [19]. Thus, the next time the DS content is updated (corresponding to a new window location) non-dominated exemplars can be prioritized for retention. Performance was compared to that of an Adaptive Naive Bayes (ANB) classifier that made similar assumptions regarding how to sample exemplars from the stream under a label budget [18].

In this work we undertake a through re-evaluation of the relation between DS updating and training epochs of GP. The hypothesis pursued here is that more than one generation may be performed per update to DS content without changing the label budget or provoking symptoms of over-learning. This is particularly important when attempting to support a tapped delay line (TDL) interface to the stream. Specifically, utilizing a delay line implies that when classifying exemplar $\vec{x}(t)$, access to a sequence of lagged instances is supported or: $\vec{x}(t-l\tau), ..., \vec{x}(t-2\tau), \vec{x}(t-\tau), \vec{x}(t)$ where l is the depth of the delay line and τ is the skip size.[1] However, each $\vec{x}$ is a vector of d

[1]Note that the label information is limited to that of $\vec{x}(t)$ alone.

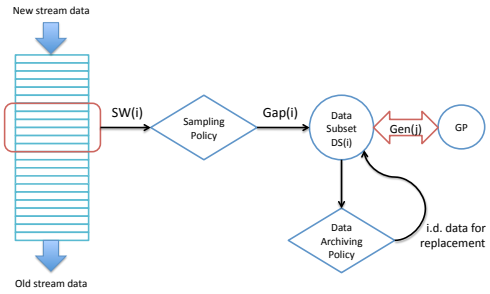

Figure 1: StreamGP architecture. $Win(i)$ denotes the location of the non-overlapping window interface to the stream. A sampling policy selects $Gap(i)$ exemplars from $Win(i)$ without prior knowledge of label information. $DS(i)$ is the data subset (with labels) against which GP training epochs are performed. The archiving policy defines which exemplars are retained as $i \rightarrow i+1$

attributes. Thus, the TDL implies that each exemplar is a matrix of $d \times l$ attributes. On the one hand this provides a mechanism for capturing temporal properties potentially useful for characterizing exemplar t. Conversely, the dimensionality of the input space has now undergone a significant increase, potentially making the task of learning appropriate models that much more difficult.

The final aspect developed by this work is with respect to benchmarking. In particular, results for streaming classification tasks are generally expressed in terms of the prequential error metric [12, 4]. Such a metric is incrementally estimated and describes classifier performance as a trajectory through the stream. The most significant drawback of such a metric, however, is that it represents an accuracy metric. Unlike approaches to offline batch performance evaluation, it is not possible to control the distribution of exemplars throughout the (stream) data set. Hence, local regions of the stream are very likely to be imbalanced, leading to low levels of class mixing and potentially degenerate classifier behaviours. Such behaviours are not identified by accuracy style metrics typically assumed for benchmarking streaming classifiers. In this work we therefore make use of an online definition for the class-wise detection metric [14].

2 Related Work

Classification under streaming tasks has been addressed from the perspective of online and ensemble learners for a considerable period of time [14, 16, 11, 10]. However, comparatively few works have proposed explicitly evolutionary computation (EC) frameworks for streaming data classification. Most EC effort has been placed on dynamic optimization tasks where the objective is to accurately track movement in multi-modal optimization tasks [6]. As such, label information is freely available and there is a much tighter coupling between representation space and search space. Several GP researchers have considered the case of evolving models against a completely labelled stream [9, 2, 5, 3]. However, other than [19], only non-evolutionary ML researchers have considered the case of explicitly developing models under label budgets e.g., [15, 17].

3 StreamSBB

EC frameworks proposed for dynamic optimization tasks in general make use of several key components of which support for evolvability, diversity and memory appear to be critical [14, 6]. In this work we are interested in further developing a recently proposed framework for applying GP to streaming classification tasks under label budgets [19]. StreamSBB assumes a symbiotic bid-based (SBB) formulation for GP, thus solutions are coevolved into teams of programs [7]. Such modularity is necessary in non-stationary tasks to provide the basis for addressing the evolvability / plasticity requirement [14]. Moreover, multi-class classification is a natural artifact of the SBB architecture. Figure 1 summarizes the framework assumed for applying GP to non-stationary streaming tasks.

StreamSBB assumes a non-overlapping window as the generic interface to the stream ($Win(i)$). For each unique window location (i), Gap exemplars are sampled with uniform probability *after which* their corresponding label is requested. Hence, a label budget is enforced. A Pareto archiving policy with an age / diversity maintenance heuristic prioritizes Gap exemplars for replacement from the Data Subset (DS) [19, 1]. At this point ρ generations are performed where previous work assumed $\rho = 1$. As noted in the introduction, such a constraint impacts on the capacity of StreamSBB to react to changes in the stream. Note also that for $\rho > 1$ there is no change to DS

Table 1: Benchmarking dataset properties. d denotes dimensionality, N refers to the total exemplar count, k reflects class count of datasets, 'Change' denotes the % of the stream in which a label change occurs.

Stream / Dataset	d	N	k	≈Class Distrib. (%)	Change
Gradual Concept Drift (drift)	10	150,000	3	[16, 74, 10]	40%
Sudden Concept Shift (shift)	6	6,500,000	5	[37, 25, 24, 9, 4]	73%
Electricity Demand (electricity)	8	45,312	2	[58, 42]	15%
Churn Detection (churn)	16	1,669,593	2	[91, 9]	11%

content. Only when the position of the non-overlapping window to the stream shifts to the next 'chunk' of stream data is there an update to DS content. Hence, the index (i) for window location, label requests, and updates to DS content are all the same.

On the face of it, this is a relatively minor modification to the StreamSBB framework. However, as will become apparent from Section 5 the selection of $\rho > 1$ has a significant impact on the overall performance of the algorithm. Moreover, the significance of this is all the more pressing when classification of exemplar t is performed with reference to a tapped delay line (TDL) describing a sequence of lagged instances from the stream, i.e. a potentially more descriptive representation (see discussion from Section 1). The input now takes the form of a $d \times l$ matrix (# attributes by #TDL taps), thus GP needs to rationalize what specific instances to utilize while the stream continues to pass by.

Finally, anytime operation (as in predicting labels for stream content) is supported using the current $DS(i)$ content[2] to define a 'champion' individual. A class-wise detection rate metric is estimated across $DS(i)$ content for the non-dominated SBB teams alone (Section 4.3, Equ. (2) estimated across $DS(i)$). The latter constraint providing some robustness to selecting degenerate SBB teams. That is to say, $DS(i)$ content is a function of classes sampled from the stream under a finite label budget, thus could potentially only consist of exemplars from a single class.

4 Experimental Methodology

4.1 Streams / datasets

Four datasets are used for the purposes of benchmarking, two of which are artificially generated and two represent real world tasks that potentially contain non-stationary properties. The reason to include two artificially generated streams is that with real-world datasets the degree of non-stationarity is unknown. Thus, artificially generated streams enable different forms of concept change to be embedded.

The artificial data streams[3] are denoted "Gradual Concept Drift" and "Sudden Concept Shift" and the real-world datasets are "Electricity Demand" [13] and "Churn Detection". Electricity Demand has frequently been employed for streaming data benchmarking tasks [14] whereas the second represents an online video gaming churn prediction task. The basic properties of the datasets are summarized by Table 1 in which the last column (or 'Change') captures the frequency of label changes through the stream.

Gradual Concept Drift stream [8]: Hyperplanes are defined in a 10-dimensional space. Initial values of the hyperplane parameters are selected with uniform probability. This Dataset has 150,000 exemplars and every $1,000$ exemplars, half of the parameters are considered for modification with a 20% chance of change, hence creating the gradual drift of class concepts. Class labels are allocated as a function of hyperplanes exceeding a class threshold.

Sudden Concept Shift stream [20]: The stream is created 'block-wise' with 13 blocks and each block consists of 500,000 exemplars. Consider a concept generator tuple of the form: $\langle C1\%, C2\% \rangle$ where C1 and C2 represent two independent rule sets defining 5 class tasks. A stream is now defined in terms of the transition of exemplars from 100% $C1$ to 100% $C2$ in 10% increments: $\langle 100, 0 \rangle$, $\langle 100, 0 \rangle$, $\langle 100, 0 \rangle$, $\langle 90, 10 \rangle$, $\langle 80, 20 \rangle$, ... $\langle 0, 100 \rangle$. A uniform p.d.f. is used to determine exemplar sequencing in each block.

Electricity Demand characterizes the rise and fall of electricity demand in New South Wales, Australia, using consumption and price information for the target and neighbouring regions [13]. As such it is a two class dataset

[2]The only source of labelled data.

[3]`http://web.cs.dal.ca/~mheywood/Code/SBB/Stream/StreamData.html`

Table 2: Stream dataset max. window count (i_{init}) and label budgets (LB)

Stream / Dataset	S_{max}	i_{max}	LB
Gradual Concept Drift (drift)	150,000	500	6.7%
Sudden Concept Shift (shift)	6,500,000	1,000	0.3%
Electricity Demand (elec)	45,312	500	22.1%
Churn Detection (churn)	1,669,593	1,000	1.2%

Table 3: Generic SBB parameters. Symbiont population varies dynamically, hence no size parameter is defined. SBB assumes a 'breeder' model of evolution in which M_{gap} hosts are removed per generation [7].

Parameter	Value	Parameter	Value
Data Subset (DS) size	120	Host pop. size (M_{size})	120
Prob. symbiont deletion (p_d)	0.3	Prob. symbiont addition (p_a)	0.3
Prob. action mutation (μ_a)	0.1	Max. symbionts per host (ω)	20
Host pop. gap size (M_{gap})	60	Data Subset gap size (Gap)	20

(demand will either increase or decrease relative to the previous period).

Churn Detection determines the loyalty of a player toward the online video game he/she is playing. There are 16 features describing each turn of the game as well as some player-related features. The label indicates whether the player will (or not) churn within the upcoming 24–48 hour time window (horizon). The set is quite unbalanced with about 91% being class 0 (will not churn) players who will keep playing after the time window and only about 9% being class 1 (will churn), i.e. most of the time players do not churn within the 2 day period.

4.2 Parameterization of GP

For a test stream of S_{max} exemplars a non-overlapping window of length S_{max}/i_{max} exemplars is assumed where i_{max} are the number of window locations. The remainder of the stream passes through at a constant rate. The window content defines the pool from which the new $Gap(i)$ training exemplars are sampled and labels requested (Figure 1).

Model initialization is performed using the first S_{init}% of the stream during the first i_{init}% of generations. Given that the interface to the stream assumed by StreamSBB is a non-overlapping window, then this defines the initial window length and implies that i_{init}% of the generations are performed against this window location. Thereafter, the sliding window advances at a fixed rate through the stream. Both S_{init} and i_{init} parameters are set to 10 percent.

Label budget is the ratio of exemplars whose labels are requested to the total stream length, or:

$$label\ budget\ (LB) = \frac{i_{max} \times |Gap|}{S_{max}} \tag{1}$$

In other words only $i_{max} \times |Gap|$ exemplars are sampled in a stream of length $S_{max}(\equiv N)$; whereas the total number of generations performed is: $i_{max} \times \rho$. Only $|Gap| = 20$ exemplars are added to $DS(i)$ (by the Sampling Policy) at each window location, $Win(i)$; hence, the label budget in the specific case of the concept shift data set would be: $LB = \frac{1{,}000 \times 20}{6{,}500{,}000} \approx 0.3\%$

Given the variation in stream lengths of the benchmarking datasets (Table 1), different parameterizations for i_{max} will be assumed per dataset (summarized by Table 2). Note that i_{max} is taken to include the pre-training budget i_{init}%. Table 3 summarizes the remaining generic SBB parameter settings assumed in this study e.g., population size, variation and selection operator frequencies.

Parameterization of TDL ($\vec{x}(t - l\tau), ..., \vec{x}(t - 2\tau), \vec{x}(t - \tau), \vec{x}(t)$) defines the length ($l$) and skip size ($\tau$) assumed or $tapSize$ and $tapSkip$ respectively. The range used for $tapSize$ is $[0, ..., 7]$, and $tapSkip$ is defined as $[1, 2, 4, 8, 16]$. Naturally, setting $tapSize = 0$ implies that classification is performed relative to $\vec{x}(t)$ alone. Small values for $tapSkip$ imply more locality (greater resolution) whereas larger values increase the range covered by the TDL albeit at a lower resolution.

DS oversampling (or simply referred to as **oversampling**) reflects the ability of StreamSBB to decouple the rate at which GP training epochs are performed from the rate at which the data subset content is updated (Section 3).

The degree of oversampling is parameterized as follows: $\rho \in \{1, 2, 5\}$; where $\rho = 1$ implies one GP training epoch per data subset.

4.3 Detection rate for stream data

An online class-wise **detection rate** provides the basis for incrementally estimating detection rate throughout the stream while being robust to class imbalance (unlike accuracy or error style metrics) [14]. This is particularly important under streaming data situations as models are updated incrementally and therefore sensitive to the distribution of current window content (typically a skewed distribution of classes even when the overall class distribution is balanced). The incremental class-wise detection rate can be estimated directly from stream content as follows:

$$DR(t) = \frac{1}{C} \sum_{c=[1,..,C]} DR_c(t) \text{ where } DR_c(t) = \frac{tp_c(t)}{tp_c(t) + fn_c(t)} \tag{2}$$

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

4.4 Comparator model

The comparator classifier is documented in a recent study of streaming data classification under label budgets and drift detection [17], and has been made available in the Massive Online Analysis (MOA) toolbox.[4] Specifically, the **Adaptive Naive Bayes (ANB)** classifier with budgeted active learning and drift detection. The 'random' active learning strategy was selected as it provided the baseline in [17] and is closest to the stochastic sampling policy adopted in this work. Active learning with budgeting is managed under a random exemplar selection policy in which stream data is queried for labels with frequency set by the budget parameter.

5 Results

Benchmarking is performed in three phases in order to assess: 1) the contribution from tapped delay lines, 3) the role of oversampling, and 3) the performance relative to the adaptive Naive Baysian framework. All StreamSBB results represent the average of 50 runs, thus any StreamSBB performance curve is an averaged curve.

5.1 Experiments

Tap delay line (TDL) experiment: As per Section 4.2, between 1 and 7 historical instances can be attached to the current instance (*tapSize*). Such instances can be offset by 1 to 16 instances far from current instance (*tapSkip*).

Figure 2 illustrates how increasing tap size from 1 to 3 to 7 collectively decreases the detection rate of StreamSBB under the concept shift stream regardless of tap skip. The original StreamSBB without TDL provides an indication of the baseline performance (black dashed curve). The other datasets follow a similar trend of diminishing detection rate while increasing tap size. Varying the tap skip value provides greater history in the samples retained within the delay line. The electricity demand dataset was the only data set to respond particularly favourably to increases to skip size (page limit precludes a supporting figure).

DS oversampling experiment: Figures 3 and 4 illustrate the impact of oversampling in terms of detection rate curves for concept drift and shift streams. Higher detection rates are now maintained throughout the stream. The higher rate of oversampling appears to be preferable throughout. Further increases to the oversampling (say to a factor of 10) has only marginal positive effects (results not shown for clarity). To confirm the significance of the difference between each pair of curves, the nonparametric Mann-Whitney U test[5] of the null hypothesis is assumed (i.e. does not assume a normal distribution for data). A significance level of 0.01 is assumed and report the p-value of the test when the null hypothesis is rejected ($h = 1$) in Table 4. In all cases the higher rate of oversampling is preferred. Moreover, applying a Bonferroni Correction of $0.01/3$ to the p-values does not change this conclusion. Results for the real-world datasets were also positive and will be reported later when we compare with the ANB framework for streaming classification.

[4] MOA prerelease 2014.03; `http://moa.cms.waikato.ac.nz/overview/`

[5] Also referred to as the Wilcoxon rank-sum test or Wilcoxon-Mann-Whitney test.

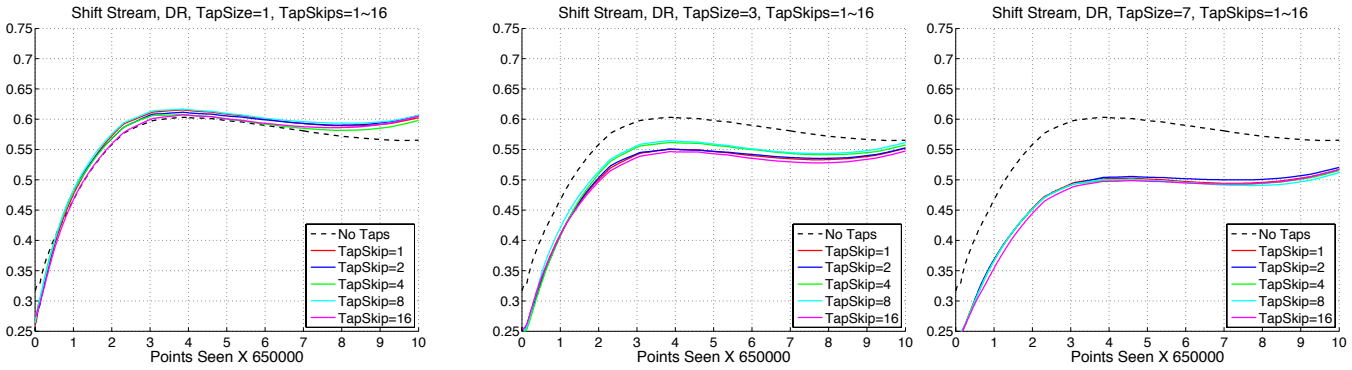

Figure 2: Detection rate of StreamSBB for concept shift stream. The dashed curve is the case of no TDL (i.e. original StreamBB) and the solid colored curves are StreamSBB using TDL with different *tapSize* and *tapSkip* parameters.

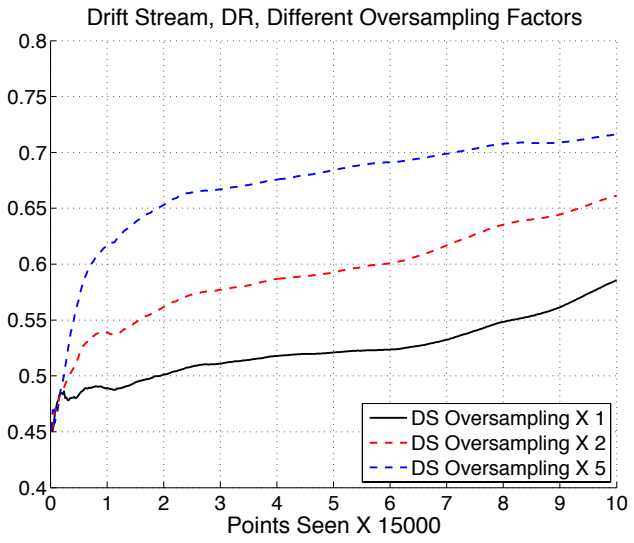

Figure 3: *DR* on **drift** stream. Black: default sampling; Red: $\times 2$ oversampling; and Blue: $\times 5$ oversampling.

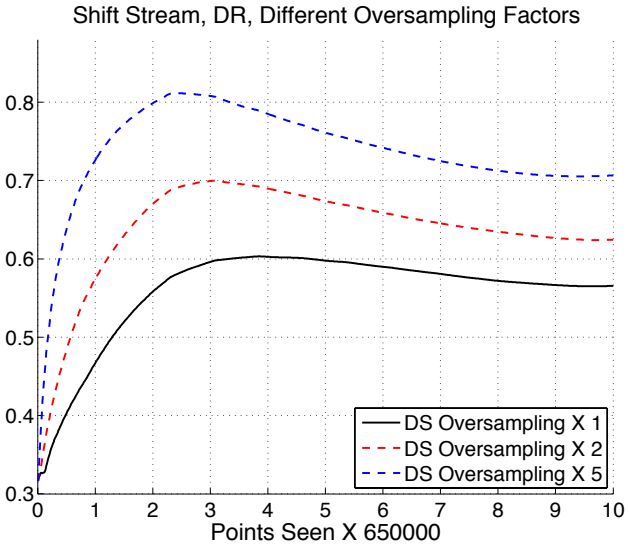

Figure 4: *DR* on **shift** stream. Black: default sampling; Red: $\times 2$ oversampling; and Blue: $\times 5$ oversampling.

Combined TDL and DS oversampling: Figures 5 and 6 show detection rate curves corresponding to different configurations of StreamSBB with or without TDL and/or DS oversampling. Note that for clarity the tap skip is fixed to 16 for all curves using TDL. Choosing other tap skip values returns almost identical curves for all but the electricity demand dataset.

The black solid curve is the original StreamSBB baseline before using TDL or DS oversampling. The three red curves show the diminishing trend of detection rate as tap size increases (oversampling disabled). The two blue curves show the detection rate curves when an oversampling rate of 2 and 5 are applied (TDL disabled). The black dashed curve is the detection rate curve when a tap delay line with tap size 1 and DS oversampling of rate 5 is enabled. We call this the *optimal StreamSBB* configuration. It is evident that detection rate improves once both properties are enabled. Table 5 reports the p-values of the Mann Whitney U test when comparing the optimal StreamSBB (using tap size 1 and oversampling rate of 5) with the original StreamSBB (disabling TDL and DS oversampling) under 0.01 significance level.

In summary, the above experiments demonstrated that a tap size of 1 outperformed a tap size 3 and 7, and the DS oversampling rate of 5 yielded statistically significantly better results than an oversampling rate of 2. A tap skip of 16 was used to take advantage of farthest historical instance for electricity dataset. Under these parameter settings the detection rate for the original StreamSBB, optimal StreamSBB and the Adaptive Naive Bayes (ANB) classifier are displayed in Figures 7 to 10.

It appears that ANB has problems when there are sudden changes to the content of the stream (Figure 8),

Table 4: Mann Whitney U test p-values for comparing different pairs of curves. $\times 1$, $\times 2$ and $\times 5$ define 1, 2 and 5 generations per DS update respectively.

Stream / Dataset	$\times 1$ vs. $\times 2$	$\times 1$ vs. $\times 5$	$\times 2$ vs. $\times 5$
Gradual Concept Drift	3.84e-9	5.02e-17	6.30e-13
Sudden Concept Shift	3.92e-8	1.37e-17	9.92e-13

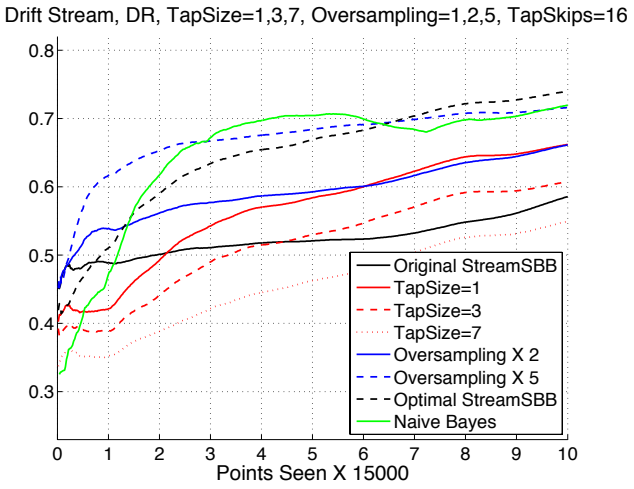

Figure 5: *DR* on **drift** stream. TDL and oversampling experiments.

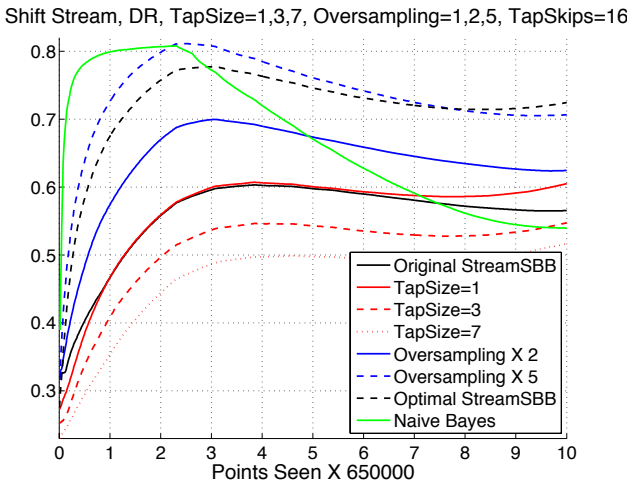

Figure 6: *DR* on **shift** stream. TDL and oversampling experiments.

Table 5: Mann Whitney U test p-values for comparing Optimal StreamSBB and Original StreamSBB detection distributions.

Data set	Drift	Shift	Electicity	Churn
p-value	8.46e-18	1.08e-17	7.07e-18	7.06e-18

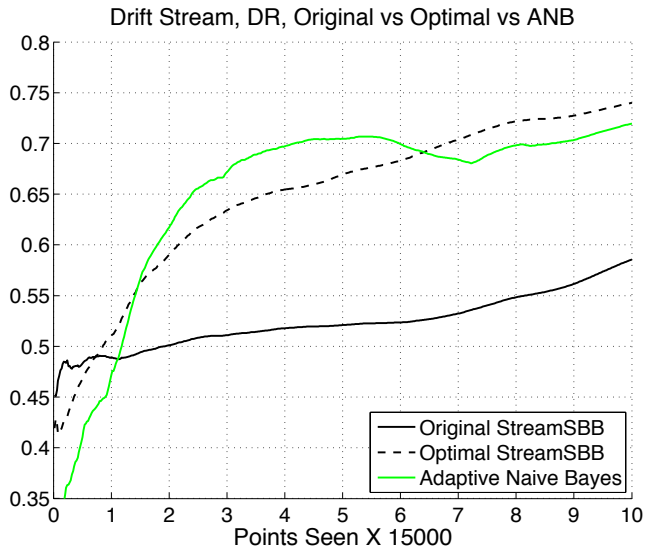

Figure 7: DR curves for Concept Drift

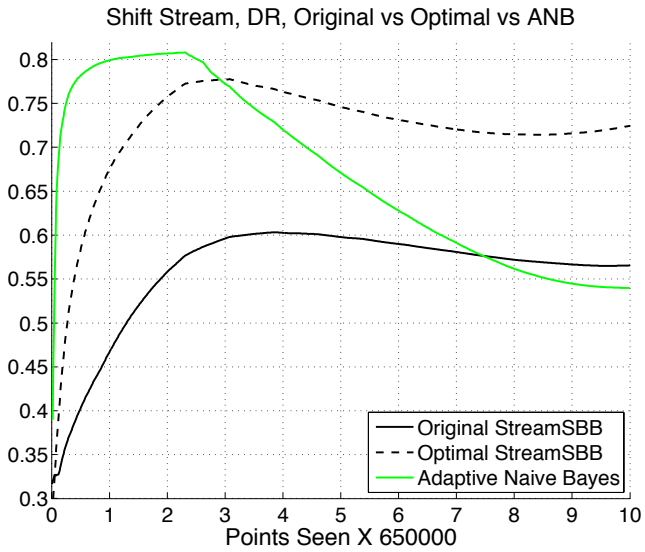

Figure 8: DR curves for Concept Shift

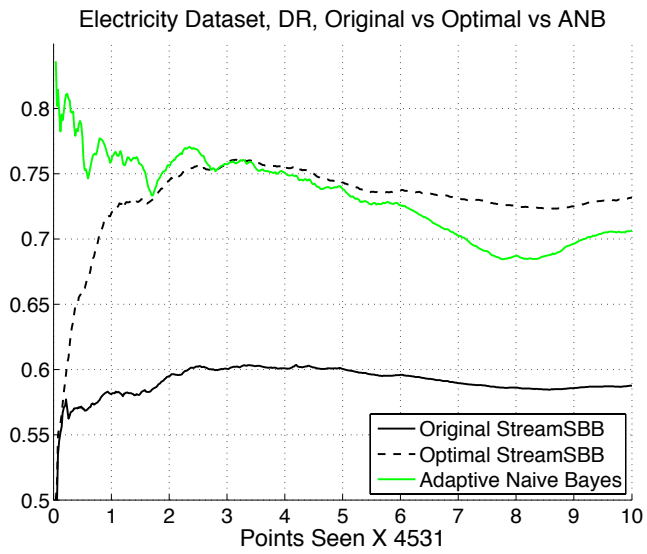

Figure 9: DR curves for Electricity Demand

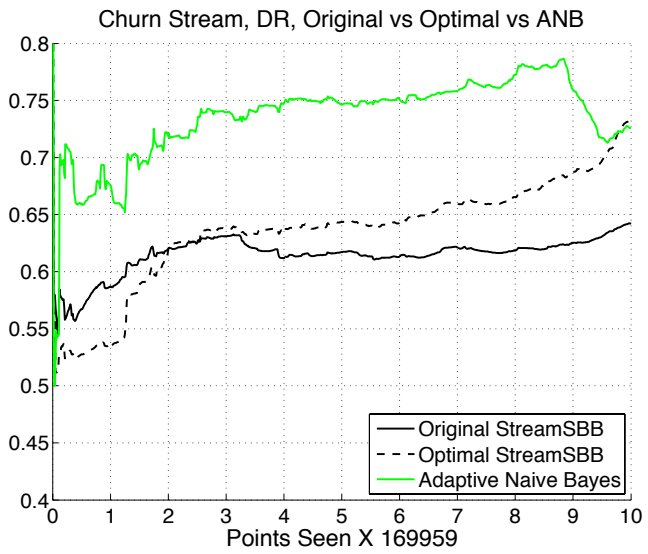

Figure 10: DR curves for Churn Detection

whereas both algorithms are effective under the drift stream (Figure 7). In all cases ANB is also able to return better detection rates much faster than StreamSBB. However, during the course of the stream ANB performance decays whereas StreamSBB performance generally continues to climb, ultimately resulting in StreamSBB reaching or exceeding the performance of ANB.

6 Conclusion

The StreamSBB framework has been revisited to support a TDL input representation. This is of fundamental importance when building classifiers for streaming applications. However, the dimensionality of the input space also undergoes a significant increase. Given that classifiers are built incrementally as the stream passes, it is necessary to make sure that the 'rate of evolution' is significantly higher than the rate at which the subset of labelled data 'turns over'. This is the role of DS oversampling ($\rho > 1$). Without this, StreamSBB performance is 10 to 15% worse than originally configured. Moreover, this is performance as measured in terms of online class-wise detection rate, thus unaffected by merely improving performance on the majority class. This is the first time that such a metric has been demonstrated under empirical conditions (previous formulations being limited to offline evaluation scenarios).

Acknowledgments. The authors gratefully acknowledge support from NSERC Discovery and CRD programs (Canada) and RUAG Schweiz AG (Switzerland) while conducting this research.

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 885–892, 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] M. Behdad and T. French. Online learning classifiers in dynamic environments with incomplete feedback. In *IEEE Congress on Evolutionary Computation*, pages 1786–1793, 2013.

[4] 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.

[5] A. Cervantes, P. Isasi, C. Gagné, and M. Parizeau. Learning from non-stationary data using a growing network of prototypes. In *IEEE Congress on Evolutionary Computation*, pages 2634–2641, 2013.

[6] I. Dempsey, M. O'Neill, and A. Brabazon. *Survey of EC in dynamic environments*, chapter 3, pages 25–54. 2009. In ([**?**]).

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

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

[9] G. Folino and G. Papuzzo. Handling different categories of concept drift in data streams using distributed GP. In *European Conference on Genetic Programming*, volume 6021 of *LNCS*, pages 74–85, 2010.

[10] J. Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

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

[12] J. Gama, R. Sebastião, and P. Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[13] M. Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, University of New South Wales, 1999.

[14] M. I. Heywood. Evolutionary model building under streaming data for classification tasks: opportunities and challenges. *Genetic Programming and Evolvable Machines*, (DOI 10.1007/s10710-014-9236-y), 2015.

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

[16] R. Polikar and C. Alippi. Guest editorial: Learning in non-stationary and evolving environments. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):1–3, 2014.

[17] 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.

[18] I. Žliobaitė and B. Gabrys. Adaptive preprocessing for streaming data. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):309–321, 2014.

[19] 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 Genetic and Evolutionary Computation Conference: ECBDL Workshop*, pages 1287–1294, 2014.

[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.