Partial Drift Detection Using a Rule Induction Framework

Damon Sotoudeh
York University, Toronto, Canada
4700 Keele Street
Toronto, Canada
damon@cse.yorku.ca

Aijun An
York University, Toronto, Canada
4700 Keele Street
Toronto, Canada
aan@cse.yorku.ca

ABSTRACT

The major challenge in mining data streams is the issue of concept drift, the tendency of the underlying data generation process to change over time. In this paper, we propose a general rule learning framework that can efficiently handle concept-drifting data streams and maintain a highly accurate classification model. The main idea is to focus on partial drifts by allowing individual rules to monitor the stream and detect if there is a drift in the regions they cover. A rule quality measure then decides whether the affected rules are inconsistent with the concept drift. The model is accordingly updated to only include rules that are consistent with the newly arrived concept. A dynamically maintained set of instances deemed relevant to the most recent concept is also kept at memory. Learning a new concept from a larger set of instances reduces the variance of data distribution and allows for a more accurate, stable classification model. Our experiments show that this approach not only handles the drift efficiently, but it also can provide higher classification accuracy compared to other competitive approaches on a variety of real and synthetic data sets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*data mining*

General Terms

Algorithms

Keywords

Concept Drift Detection, Rule Induction, Classification

1. INTRODUCTION

One of the challenges in data stream classification is that the underlying data generation process of a stream tends to change over time, a phenomenon called concept drift. A model learned from an earlier part of stream loses its classification accuracy upon the arrival of new instances that exhibit concept drift. As a result, any classification algorithm that is to be applied to data streams must be adjusted to effectively detect the concept drift when it occurs, and efficiently update the classification model to reflect the new concept.

A common, and perhaps most intuitive, metric of detecting drifts is classification accuracy. A drop in the classification accuracy of the current classifier on the new data signals a possible drift, which in turn requires updating the current model. Since classification accuracy is dependent on the structure of the entire model, to observe a noticeable drop in accuracy, a drift should sufficiently affect a major part of the model. However, drifts are likely to be partial, affecting only particular regions of the data. Such partial drifts, especially if small in magnitude, may go unnoticed by model-level metrics like classification accuracy. Instead, it is beneficial to have a metric that can monitor data on a particular region, and sensitive enough to detect changes in that region. Such metric allows for detection of partial as well total drifts.

Upon detecting a concept drift, the classification model must be updated to reflect the information available from the new instances. To make the effect of the recent instances more prominent, it is important to forget some of the older, irrelevant instances. The commonly utilized sliding window strategy [14, 10] imposes a sliding window on the data stream and provides the content of the window to the classifier. As the window moves forward, new instances are inserted into and old instances are dropped out of the window. The advantage of this method is that the classification algorithm is always provided with the most *recent* data instances, reflecting the most recent concept. It is not trivial to decide on the number of instances a window should contain [14, 10]. If the window is small, the learned classification model becomes specific and sensitive to noise or data variation within the same concept. If the window size is large, the classification model becomes generalized and provides robustness to noise but may show reluctance to adapting to new concepts. Furthermore, a window based strategy does not take the relevance of instances into account. Old instances are not necessarily irrelevant to the current concept, and may still provide useful information for the new concept. As a result, a forgetting strategy that forgets instances solely based on their time stamps is inferior in performance to a strategy that takes the relevancy of those instances into account.

In light of these observations, we provide a novel approach for handling concept drifts using a rule induction framework. In this framework, instead of monitoring the incoming

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

stream from a classifier perspective, we move down to individual rules, and use them for detection of partial changes in the stream. Each rule is further augmented with a measure of quality that decides if an affected rule has become inconsistent with respect to the new concept, and the model maintains only the rules that are consistent with the current concept. Our method also aims to provide the classification algorithm with the instances that are considered both *relevant* to the current concept. Inclusion of recent instances reduces the variance of data distribution and allows for construction of a more stable, accurate classification model.

The contributions of this paper are as follows:

- We propose a novel rule learning framework for detecting and handling partial drifts in data streams.
- We propose a few metrics that can be used to detect drifts in local regions of data.
- We use the concept of rule quality and introduce how it can be used to determine rule inconsistency on emerging concept.
- We further use the rule qualities and the rules themselves to judge the relevance of past instances to the current concept to provide the classification algorithm with the most relevant and recent instances.
- Using an extensive set of experimental studies, we show that our method can quickly react to concept drifts, construct highly accurate classification model, and offer time and space efficiency.

Paper Organization. In section 2 we look at some related work. Section 3 formally defines concept drift, and introduces our proposed measures of detecting drifts. Section 4 discusses our instance forgetting strategy, and section 5 describes the model updating procedure, followed by the general description of the framework section 6. An extensive set of experiments is provided in section 7. Finally, we conclude the paper in section 8.

2. RELATED WORK

CVFDT [7] is a stream classification algorithms that grows alternative tree branches when a sub-tree becomes out of date. Once the classification accuracy of the new sub-tree surpasses that of the outdated one, the former replaces the latter.

A powerful classification technique is the Ensemble approach [12], which is applied to sequential chunks of data. A classifier is learned separately on every data chunk, and each classifier is then assigned a weight inversely proportional to its expected classification error. Classification is done by a voting process where each classifier decides on the label of testing instances and a weighted score decides the class.

RePro [15] maintains a history of concepts in memory and examines the accuracy of the them on the emerging instances. If a high performing model on the new data can be found, then the current model is replaced with it. Otherwise, a new model is learned based on the new data. Although many concepts are historically stored, RePro does not use their knowledge during the classification process. It seems, considering the data is stored anyway, a more collaborative approach using all the concepts would provide better results.

The KL-divergence method [5] is a statistical approach that compares two separate batches of data, uses the KL-divergence measure to calculate the distribution distance between them, and applies bootstrapping to decide if the change is statistically significant. A statistically significant change suggests occurrence of a concept drift. The entropy based detection method [11] calculates the change in entropy of two windows of instances. If the change exceeds certain thresholds, then a drift is detected.

3. CONCEPT DRIFT DETECTION

3.1 Definitions

Concept drift is defined as a change in the underlying data generation process. In the context of classification, concept drift is the change in statistical properties of the target variable, which the model is trying to predict, over time [1]. In this context, the term *concept* refers to the quantity we aim to predict.

Let $\mathbf{x}$ be an instance in an m-dimensional feature space and c_i a class label where $i \in \{1, 2..., k\}$. The objective of classification is to find c_i that maximize $P(c_i|\mathbf{x})$. Thus, in classification learning, concept drift can be considered as a change in $P(c_i|\mathbf{x})$. In [14], such a change is referred to as *real drift*, while a change in $p(\mathbf{x}|c_i)$ (i.e., the distribution change in the input) is referred to as *virtual drift*.

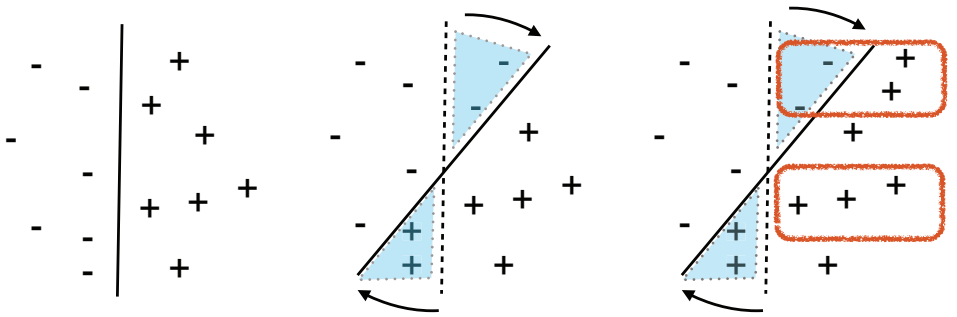
Most of the concept drift detection methods detect drifts by measuring the changes in the whole instance space. That is, when measuring the change in $P(c_i|\mathbf{x})$, $\mathbf{x}$ ranges over the entire feature space $\Re^m$, where m is the dimensionality of the feature space. We refer to such a drift detection method as a *global drift detection method*. It is common, however, that a change in $P(c_i|\mathbf{x})$ occurs only in a subspace of $\Re^m$. Figure 1(a) illustrates an initial class boundary in a 2-dimensional feature space. After a period of time, the concept undergoes a drift and its class boundary rotates clock-wise to the position of the solid line in Figure 1(b). Only in the shaded regions $P(c_i|\mathbf{x})$ has changed; in the other regions the class labels have not changed. We refer to such changes in local regions of the instance space as *partial concept drifts*.

3.2 Partial Drift Detection Method

A limitation of global drift detection is that if the data undergo a partial drift in a small region, a global method may not be sensitive enough to detect it. Also, a global method may not be able to identify the regions where the change takes place and thus unable to update the model accordingly.

To detect partial drifts, we need to look into local regions of the instance space. One way to do so is to partition the instance space into subspaces and apply an existing global method to each of the subspaces. Such a strategy requires prior knowledge of how the instance space should be partitioned. A coarse partition may not offer much of the benefits of partial detection, while a fine partition may be too sensitive to noise and data variation. To overcome these problems, we propose a partial drift detection method based on a rule induction framework which we describe below.

1. Apply a rule induction method to the current available data (e.g., the first chunk of data) to learn a set of classification rules.



(a) Initial position of the hyperplane. (b) Rotated hyperplane and partial drift. (c) Changes in rule accuracy after drift.

Figure 1: Formation of partial drift and its effect on rules.

2. When a new data chunk is available, detect partial drifts as follows:

   (a) For each rule whose coverage is over a user-specified threshold, apply a drift detection measure (described later in Section 3.3) to detect changes in the local instance space covered by the rule.

   (b) If no rule has coverage over the user-specified threshold, apply the same measure to each rule whose coverage is over the 90th percentile of the rule coverage in the model to detect drifts in the local region covered by the rule.

In this procedure, only rules with good coverage are used to identify regions for local drift detection. The reason for such a choice is that changes in a small region may well be due to noise or data variation and thus are not reliable indicators of concept drifts. In addition, rules with good coverage usually describe the concept in the data better than the rules that cover few examples. A significant change of class distribution in the region covered by such a rule is a better indicator that the concept is changing. The coverage threshold can be user-specified (such as 10%), or a high percentile of the rule coverage values. We use a user-specified threshold first. If none of the rules has coverage over the threshold (meaning all the rules have small coverage), we choose all the rules whose coverage is over the 90th percentile. The reason for this second threshold is to choose rules with relatively high coverage from low-coverage rules; otherwise, no rules can be used for drift detection. The choice of the 90th percentile is based on our experimental results.

Figure 1(c) illustrates how a rule can be used to detect partial drifts. A rectangle represents the region covered by a rule learned before the drift occurred. It is obvious that the class distribution in the top rectangle has changed significantly after the drift and focusing on such a region can effectively detect such a partial drift, while with a global method the change may not be considered significant. The use of rules in our method allows for fine-level detection of partial concept changes in the underlying data distribution. It may be incorrectly concluded that this drift detection method is limited only to partial drifts. Quite the contrary is true, however. A global drift can be essentially thought as the accumulated effect of a large number of partial drifts. A partial drift affects small regions of the data and consequently only a few rules detect a drift; a global drift affects a large segment of data and consequently many more, possibly all, rules detect a drift. In section 7, we will show that this framework is capable of detecting both types of drift. More importantly, it is able to determine regions that undergo a drift.

3.3 Measures for Partial Drift Detection

To detect a drift in the region covered by a rule, a metric is needed to measure the distribution change of the target variable between the old and new data. In this section, we introduce several measures that can be used for this purpose. The target variable in this context is the class variable c. The goal is to measure the distribution change of c in the region covered by rule R, i.e., $P(c|R)$.

3.3.1 χ^2 test

Pearson's chi-square test is a non-parametric statistical test used to determine if there is a significant difference between the expected frequencies and observed frequencies in one or more categories. We can use it to determine whether the class distribution of instances covered by a rule in the new data is significantly different from the one in the old data. The categories in the χ^2 test are described in the contingency table in Table 1, where c represents the class a rule R predicts, $\bar{c}$ represents the classes other than c, *old* represents the old data and *new* denotes the new data.

Table 1: The contingency table for χ^2 test.

	c	$\bar{c}$
old	$O_{1,1}$	$O_{1,2}$
new	$O_{2,1}$	$O_{2,2}$

The numbers in the cells represent the observed frequencies of the instances in each category. For example, $O_{1,1}$ is the number of positive instances covered by R in the old data. The expected frequency of each cell is estimated as:

$$E_{i,j} = \frac{\sum_{i=1}^{2} O_{i,j} \times \sum_{j=1}^{2} O_{i,j}}{\sum_{i=1}^{2} \sum_{j=1}^{2} O_{i,j}} \quad (1)$$

The χ^2 statistic is calculated as:

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}} \quad (2)$$

If the χ^2 value exceeds a critical value for one degree of freedom, the null hypothesis that the class distribution of instances covered by rule R in the new data is the same as the one in the old data can be rejected at the corresponding level of confidence. In our method, we use the confidence

level of 95%. Thus, if the χ^2 value exceeds 3.84, we conclude that the region covered by R has undergone a drift, detecting a partial drift affecting only a part of the data.

3.3.2 Entropy-based measure

Entropy can be used to measure the impurity of a data set with respect to the class membership. The entropy of the data set covered by a rule R can be calculated as:

$$H(R) = -\sum_{i=1}^{k} P(c_i|R) log P(c_i|R) \quad (3)$$

where c_i is the i^{th} class label from a total of k class labels. A change in entropy from old instances to the new ones suggests a change in the class distribution of instances covered by R. If such a change is significant, it can be said that the region covered by the rule has undergone a concept drift.

To determine whether the change is significant, we can use the *bootstrap method* to estimate the confidence interval of the entropy change between the old and new data. The bootstrapping procedure works by constructing m samples $S_{o_1} \dots S_{o_m}$ from the old data using random sampling with replacement. The size of each sample is equal to that of the original data. Similarly, m samples of the new data $S_{n_1} \dots S_{n_m}$ can be constructed. For each sample, entropy is calculated using Equation 3. For each pair of samples S_{o_i} and S_{n_i}, the difference between their entropies is calculated. If the entropy difference from the original old and new data is over the 95 percentile of the m entropy differences obtained from the samples, a concept drift is detected. In other words, the upper bound of the conference interval for the 95% confidence level is used as the change threshold for drift detection. The advantage of this bootstrap method is that the threshold is automatically determined. The disadvantage is that bootstrapping can be time-consuming when m is large and the rule covers a large number of examples. If time is a concern, a user-specified threshold can be used to replace the bootstrap method.

Note that the entropy measure may not be able to detect a concept drift in some cases. For example, assuming there are two classes in the data, if all the instances covered by the rule in the old data belong to class c_1 but all the covered instances in the new data belong to class c_2, then the entropy values in both old and new data are 0. Thus, although a drift has clearly occurred, there is no change in entropy, which means no drift is detected.

3.3.3 Rule accuracy

Classification accuracy of the learned model has been used to detect concept drifts in many global detection methods. Similarly, for partial drift detection, we can use *rule accuracy* to monitor the predictive performance of a rule. If the rule accuracy drops significantly on new data, a drift is detected in the region covered by the rule. The accuracy of a rule is the number of positive instances covered by the rule over the total number of instances covered by the rule. Similar to the entropy-based method, the bootstrap method can be used to determine a threshold for the significant drop. Alternatively, a user-supplied threshold can be used.

3.3.4 Rule quality based measures

Rule quality refers to the goodness and reliability of a rule. Many rule quality measures have been proposed to evaluate rules for classification and postpruning tasks [4]. Here we use rule qualities to also detect partial drifts. A rule with good quality provides good generalization of the data it covers. If the quality of a rule generated from previous data drops significantly when evaluated on the new data, it is a good indication that the class distribution of data in the region the rule covers has changed. In other words, a partial drift may have occurred in the region. Rule accuracy can serve as a rule quality measure. Below we introduce three additional measures.

- *Rule AUC.* The area under the ROC curve (AUC) of a rule is computed as the area of the polygon defined with the four points $(0, 0)$, (x, y), $(1, 1)$ and $(1, 0)$ in the ROC space, where x is the false positive rate and y is the true positive rate when using the rule to classify the examples it covers.
- *Logit of Rule Accuracy.* It is calculated as

$$\log \frac{P(c|R)}{1 - P(c|R)}. \quad (4)$$

- *Rule Discrimination.* It measures the extent to which a rule can discriminate between positive and negative examples [3]. It favors both high accuracy and high coverage:

$$\log \frac{P(R|c)(1 - P(R|\neg c))}{P(R|\neg c)(1 - P(R|c))}. \quad (5)$$

Similar to the rule accuracy based method, when using a rule quality based method to detect drifts, a drop threshold is needed to determine whether the drop is significant. The threshold can be either user-defined or to be determined by a bootstrap method.

4. INSTANCE FORGETTING

The sliding window technique is a straightforward approach to providing classification algorithms with the most recent of instances. Inclusion of past relevant instances into a new chunk of data in case of partial drift, however, can reduce the variance of the underlying data distribution which allows the classification algorithm to learn a more stable and accurate model. To achieve this goal, we use the rule qualities, this time to judge the relevance of old instances.

Let R be a rule in the current model. When the next data chunk D becomes available, if a drift is detected in the region covered by R, we check whether the quality of R significantly drops on the new data using one of the rule qualities described in Section 3.3. If so, it is clear that a good portion of the positive instances covered by R in the old data are inconsistent with new data, and thus can be considered irrelevant to the new concept. In this case, not only we remove R from the classification model, but we also remove the positive instances that R covers. On the other hand, rules that are not affected by the drift or rules whose qualities do not drop are consistent with the new data and positively contribute to the classification model. The instances covered by these rules are retained. The goal is to always keep a set of old instances that are considered relevant to the current concept and remove inconsistent examples.

We apply this strategy to the entire stream by consistently keeping an active set of relevant instances in memory. When a new chunk of data becomes available and a drift is

detected, the classification algorithm does not learn a model only from the newly arrived data. Instead, the new chunk is merged with the previously maintained instances that are considered relevant to it. A new model is learned from this larger data set which has less variance in data distribution, yielding a more accurate model. The merged data set serves as the set of relevant instances for the current concept.

A drawback of this merging, in certain scenarios, is the accumulation of a large number of relevant instances in memory. The possibly growing number of relevant instances can significantly affect the running time of the classification algorithm which is a major issue in stream environment. To deal with this issue, we use AVSpace [13] to compress the data by caching the sufficient statistics of relevant examples. AVSpace is a tree-based data structure that was originally designed to accelerate sequential covering algorithms for rule induction. It essentially limits the memory consumption of instances by storing the counts of instances satisfying various conditions used in rule induction. The upper-bound for memory consumption of AVSpace is $O(d^M)$ where d is the number of attributes and M is the number of values of each (discretized) attribute. It is important to note that the memory consumption is independent of the number of instances (N) available in the data stream and can thus be considered constant.

Description of the construction and structure of AVSpace is out of the scope of this paper, and instead must be referred to [13]. We suffice it to say that AVSpace is used here only for efficient maintenance of relevant instances in memory. To that end, a few changes has been made to the AVSpace structure to make it suitable for data streams. In particular, AVSpace was modified to dynamically add instances as they arrive from the data stream, and remove instances as dictated by the rules affected by a drift.

5. MODEL BUILDING AND UPDATING

Our approach is a general framework for rule learners, and any rule learning algorithm can be utilized for this approach. The learning, pruning and classification routines of a rule learner does not need to be changed but the rule learner needs to be augmented to support rule quality, which is a trivial task.

In our framework, we first learn a set of rules from the first chunk of data, and calculate rule qualities on that chunk using one of the rule quality measures described in section 3.3.4. When a new chunk of data is available, the drift detection method described in section 3.2 is used on rules with relatively high coverage. If a drift is detected on the region covered by a rule, we compute the qualities of this rule and the rules that were not used in drift detection (i.e., the rules with low coverage) on the new data. If the quality of a rule drops on the new data, which suggests the negative impact of the drift on the rule performance, the rule is removed from the model. Unaffected rules remain part of the classification model. Finally, if a drift is detected, a new model is learned from the new and past relevant data and added to the current classification model.

The strategy of keeping previously-learned consistent rules ensures that some of the past history is emphasized within the new model. If future concepts overlap with the current concept, which is the case in partial drifts, inclusion of these rules can improve the classification accuracy of the new model.

Algorithm 1 Stream Rule Learning Framework (SRLF)

Input:	A data chunk	D
	AVSpace (maintaining relevant instances)	Λ
	Rule-based classifier	C
	Drift detection method	M
	Rule coverage threshold	θ
Output:	Updated classifier	C
	Updated AVSpace	Λ

```
1: if C == NULL then
2:    insert instances of D into Λ
3:    C ← classifier learned from Λ
4: else
5:    for each rule R in C do
6:       if coverage(R) > θ then
7:          detect drift over the region covered by R using M
8:       end if
9:    end for
10:   if no rule has coverage > θ then
11:      for each rule R in C do
12:         if coverage(R) > the 90th percentile then
13:            detect drift over the region covered by R using M
14:         end if
15:      end for
16:   end if
17:   if drift is detected then
18:      for each rule R in C do
19:         if drift is detected over R or coverage(R) is below the
            threshold then
20:            Q_old ← rule quality from Λ
21:            Q_new ← rule quality from D
22:            if Q_old > Q_new then
23:               remove R from C
24:               if drift is detected over R then
25:                  remove positive instances covered by R from
                     Λ
26:               end if
27:            end if
28:         end if
29:      end for
30:   end if
31:   Insert instances of D into Λ
32:   if drift was detected then
33:      C ← C ∪ classifier learned from Λ
34:   end if
35: end if
36: return C and Λ
```

6. A RULE LEARNING FRAMEWORK

In this section, we combine the concepts introduced in Sections 3, 4 and 5 to provide a general rule learning framework for learning classification rules from data streams. We refer to this framework as Stream Rule Learning Framework (SRLF) and outline it in Algorithm 1.

The stream is processed in sequential, non-overlapping chunks. When the first chunk of data arrives, an instance of AVSpace is created and is populated with the instances in the chunk, and classification model is learned from the AVSpace (*code lines 1 to 3*). When the next chunk of data becomes available, each rule whose coverage is more than the coverage threshold utilizes a partial drift detection method such as the χ^2 test to detect drifts over the region covered by the rule (*code lines 5 to 9*). If no rule has such coverage, then all rules whose coverage are more than the 90th percentile in the current rule model are used to detect a partial drift (*code lines 10 to 16*). If a drift is detected, then the quality of each rule in the current model that detects a drift or has

low coverage is evaluated on the new data chunk (*code lines 19 to 21*). If the quality of such a rule has dropped, then the rule is removed from the classifier and further if a drift is detected over the rule, the positive instances the rule covers are removed from the AVSpace (*code lines 22 to 27*). The remaining instances in AVSpace, all deemed relevant to the new chunk, are merged with the new chunk (*code line 31*). If a drift was earlier detected, the classification algorithm is called to learn a new set of rules, which is then combined with previous high quality rules (*code line 33*). The rules are combined as the union of two sets with duplicate rules removed.

The advantage of our approach is that it does not forget instances based on their time stamps. Instead, it uses the rules and their qualities to objectively measure the relevancy of the old instances, and only forgets instances that are considered irrelevant. Inclusion of relevant instances with the newly available data provides a more consistent data set for the rule learner and results in a more stable and accurate classification model. Another advantage of this approach is that it allows for detection of partial drifts. A partial drift affects a few of the rules, and may not have a noticeable effect on the overall classification accuracy. However, by monitoring the drift at rule level, partial drifts can be easily detected. Furthermore, by focusing on rules, it is trivial to determine exactly which parts of the model, that is which rules, are inconsistent with the new data.

Assume that each data chunk has N instances and each instance has d attributes, taking M different values. There is a one time cost of $O(d^M)$ to construct the AVSpace. Drift detection on each new chunk takes $O(rdN)$ were r is the number of rules in the model. Removal of rules can be done in constant time, and inserting new instances into AVSpace takes $O(dN)$. Assume learning a classifier from AVSpace Λ takes $O(f(\Lambda))$. The overall running time on each chunk of data is $O(rdN + f(\Lambda))$. Since learning algorithms have non-linear running times, and $r, d << N$, SRLF can efficiently process the stream.

7. EXPERIMENTS

In this section, we evaluate the performance of the SRLF framework using a variety of synthetic and real life data sets. We want to see which of the drift detection measures described in section 3.3 is suitable for partial drift detection. We also want to see which rule quality formulas are good for model updating and instance forgetting. Furthermore, we compare our SRLF framework to existing stream classification algorithms, and also compare partial drift detection to global detection methods. Finally, we discuss the effect of parameters in our framework on the performance.

7.1 Performance Measures

We evaluate these approaches using both classification accuracy and drift detection sensitivity measures. Classification accuracy of a model is computed on test data that are not present during learning. Since drifts can be controlled on the synthetic data sets, it is possible to determine the sensitivity of a drift detection method, that is, if a method has correctly detected a drift, or if it has falsely reacted to the incoming stream. A model update corresponding to an actual drift counts towards the true positive rate (TPR) of the detection method and any other update counts towards the false positive rate (FPR). We also estimate how quickly an algorithm responds to a drift by calculating its detection lag. The lag of a detection is the distance (in terms of the number of data chunks) between the time a concept drift occurs and the time an algorithm detects that drift. The lag of an algorithm on a test data set is the average lag of correct detections the algorithm has on the data set. If an algorithm does not detect any true drift on a data set, we assign its lag to ∞.

7.2 Experiment Setup

We use the ELEM2 rule induction algorithm [3] as the base learner for all the methods. It is worth emphasizing that although we use ELEM2 in our experiments, any rule learner can be used by our framework. All experiments were run on a Linux machine equipped with Intel Duo 2.0GHz processor and 512MB of RAM. On the synthetic data sets, all experiments were run 20 times and the results were averaged. Each synthetic data set contains a total of 60000 instances. Unless otherwise stated, the stream was processed in sequential chunks of 1000 instances, with a coverage threshold of 10%. We later examine the effect of these parameters on model performance. 10% noise is introduced to all synthetic data sets by randomly changing the actual class labels. The significance level in all statistical tests is set to 5%. Based on the suggestions in [5] and other empirical studies, the number of bootstraps was set to 500. Finally, to obtain the best results, in accordance with [12, 9], the number of classifiers in the Ensemble approach [12] was set to 25.

7.3 Data Sets

STAGGER [14] is used to simulate concept shifts, total changes in concept descriptions. It is defined using three attributes $size \in \{small, medium, large\}$, $color \in \{red, green, blue\}$, and $shape \in \{square, circular, triangular\}$. Three blocks of data are defined as follows. In the first block, an instance is labelled 1 if $size = small \wedge color = red$. In the second block, an instance is labelled 1 if $color = green \vee shape = circular$, and in the third block if $size = medium \vee large$.

SEA [9] simulates partial overlapping drifts through four blocks of equal size, and is generated using three attributes a_1, a_2 and a_3 taking values between 0 and 10. An instance is labelled 1 if $a_1 + a_2 \leq \theta$, where θ is a user parameter. a_3 is not effective in determining the class label in this data set. By varying the value of θ for each of the four blocks, we obtain a different concept. An instance of this data set with θ values 8, 9, 7, and 10 is created.

CIRCLES [8] simulates partial, gradual drifts where an instance is labelled positive if it is inside a circle. Concept drift is simulated by changing the centre of the circle (0.2 0.5) $\rightarrow$ (0.4 0.5) $\rightarrow$ (0.6 0.5) $\rightarrow$ (0.8 0.5), and increasing its radius 0.15 $\rightarrow$ 0.2 $\rightarrow$ 0.25 $\rightarrow$ 0.3.

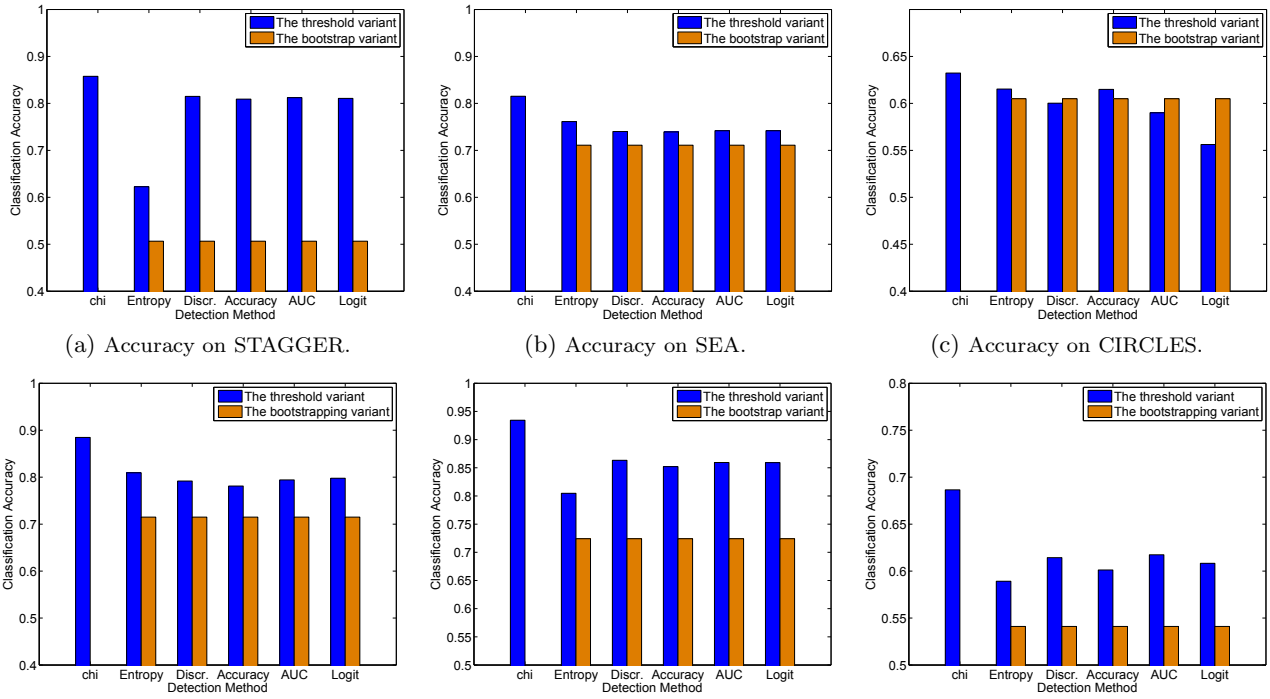
HYPERPLANE [12, 7] also simulates partial, gradual drifts by labelling instances with respect to a moving hyperplane. A d-dimensional hyperplane is defined as $\sum_{i=1}^{d} a_i x_i = b$. Only instances above the hyperplane, that is instances satisfying $\sum_{i=1}^{d} a_i x_i \geq b$ are labelled positive. Initially weights a_i are chosen randomly from [0 1] space, and b is calculated as $b = \frac{1}{2}\sum_{i=1}^{d} a_i$ so that the hyperplane divides the instance space into two sets of equal volume and equal number of positive and negative instances is generated. Instances are randomly drawn from a uniform distribution on $[0, 1]^d$.

Table 2: Sensitivity of drift detection methods with 10% change threhold.

	STAGGER						SEA						CIRCLES					
TPR	1.0	0.5	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0
FPR	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.9	1.0	1.0	0.85	0.7	0.5	0.9	1.0	1.0	0.85
LAG	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	χ^2	Ent.	Discr.	Acc.	AUC	Logit	χ^2	Ent.	Discr.	Acc.	AUC	Logit	χ^2	Ent.	Discr.	Acc.	AUC	Logit

Table 3: Sensitivity of the bootstrapping variants of detection methods.

	STAGGER					SEA					CIRCLES				
TPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FPR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LAG	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	Ent.	Discr.	Acc.	AUC	Logit	Ent.	Discr.	Acc.	AUC	Logit	Ent.	Discr.	Acc.	AUC	Logit



(a) Accuracy on STAGGER. (b) Accuracy on SEA. (c) Accuracy on CIRCLES.

(d) Accuracy on HYPERPLANE. (e) Accuracy on Network Intrusion. (f) Accuracy on Electricity.

Figure 2: Comparisons of SRLF detection methods.

Concept drifts are simulated by controlling the magnitude of change m across the dimensions that are involved in the drift for every N instances. Aside from the magnitude of change, it is also possible to control the direction of change $s_i \in \{-1, 1\}$. If there are k dimensions involved in the drift, then all weights $a_i, 1 \leq i \leq k$ are adjusted by $\frac{s_i \times m}{N}$ after the generation of each new instance. There is a 10% chance that the direction of change is reversed after N instances are generated, where s_i is replaced by $-s_i$. Once the new weights are assigned, b is re-calculated to ensure the number of positive and negative instances is equal. HYPERPLANE allows for control of the magnitude and dimensionality of the drift.

Network Intrusion [2, 15] is a real life data set exhibiting concept drift. The data set consists of separate training and testing files and each instance is labelled as either a normal connection or as one of the many attack types. The data consists of a total of 42 numeric and symbolic attributes per instance, and is gathered from a military network used to simulate network intrusions. The frequency of different types of attacks varies over time, and the data set simulates distribution changes in instance frequency. A sample of the data with 250000 instances were used.

Electricity Market Dataset [6] is another real life data set, collected from the Australian NSW electricity market. The price in this market is partially dependent on the demand and supply of electricity. The goal is to determine the relative change, that is an increase or a decrease, in pricing with respect to the average prices in the past 24 hours. The data set contains over 40000 instances and contains 6 fields explaining the electricity demand, the electricity transfer between states, etc. A detailed description is available in [6].

7.4 Comparison of Drift Detection Methods

Our SRLF framework provides the flexibility of choosing from different drift detection mechanisms such as χ^2 test or entropy-based method. Figure 2 compares these drift detection methods in terms of classification accuracy on synthetic and real data sets. As described in section 3.3, except for the χ^2 method, all the other methods have two ways to determine the change threshold: user-supplied or bootstrapped. Figure 2 shows the performance of both threshold

setting methods. Table 2 shows the sensitivity of the detection methods in terms of TPR, FPR and lag and Table 3 show the results when the bootstrap method is used.

As can be seen in Figure 2, the χ^2 based method consistently offers the best classification performance on all the data sets. All the other methods have similar performance except on STAGGER where the entropy-based method is much worse than others. The figure also shows that using the bootstrap method to determine the change threshold is not successful. Its classification performance is worse than simply using the 10% threshold on all the data sets and with all measures with only a few exceptions. The reason for such a poor performance by the bootstrap method can be explained by the sensitivity results in Table 3, which shows that the detection methods with bootstrapped thresholds are highly insensitive and do not detect any drifts on these data sets.

On the other hand, as Table 2 shows, with the exception of the entropy based method, all detection methods with the 10% change threshold correctly detect all the drifts (TPR = 1) as soon as they occur (LAG = 0). However, some methods are too sensitive that they often consider the incoming stream as drifting even when no real drift is occurring. In particular, on SEA and CIRCLES, the accuracy and AUC based approaches are so sensitive that they consider every chunk of the stream to be drifting (FPR = 1). The χ^2 approach performs better in this regard, and have relatively lower FPR without sacrificing TPR.

Two conclusions can be drawn from the above results. First, despite the soundness of the bootstrap method, it does not result in good performance. Second, amongst the detection methods, the χ^2 based approach offers not only the best classification accuracy, but also better sensitivity in terms of TPR, FPR and lag.

The entropy and rule quality based methods require a user defined threshold that determines the extent of change in entropy or drop in rule quality that is to be tolerated for drift detection. To see how the values of this threshold affects the results, Table 4 show the classification accuracy of a rule quality based detection method with different threshold values. In this experiment, rule quality measure is *rule discrimination*. Table 4 shows the best result on each data set in bold. The best result for each data set is obtained from a different threshold value, demonstrating the best threshold depends on the data set. However, the differences in accuracy are not significant among the threshold values.

One interesting observation is that on the CIRCLES data set, a very high threshold offers better results. The characteristic of the data set is such that a large number of rules with very low coverage are formed. Due to low coverage, rules are generally very sensitive, and can be easily marked for deletion from the model, resulting in instability and low classification accuracy. Using a very high drop threshold allows for maintenance of a larger number of rules in the model, and improves the classification performance.

7.5 Comparison of Rule Qualities for Model Updating and Instance Forgetting

Upon detecting a drift, SRLF uses rule qualities to determine which rules should be removed from the model and which instances should be removed from the old data. It is thus important to observe the effect of the choice of rule quality on performance. Since the result of section 7.4 suggests the χ^2 test as the best detection measure, we examine the behaviour of rule qualities in model updating and instance forgetting with the χ^2 test for drift detection. Figure 3 shows the result. In general, the choice of rule quality does not significantly affect the classification accuracy. This is because, if the value of one measure drops, it is very likely that the values of other measures drop too, though the extent of the drop varies slightly from one quality measure to another, resulting in small difference in performance. Consequently, any of these measures of rule quality can be used for model updating and instance forgetting without significant impact on the model performance. In general, however, rule discrimination that takes both accuracy and coverage into account performs best in most of the datasets.

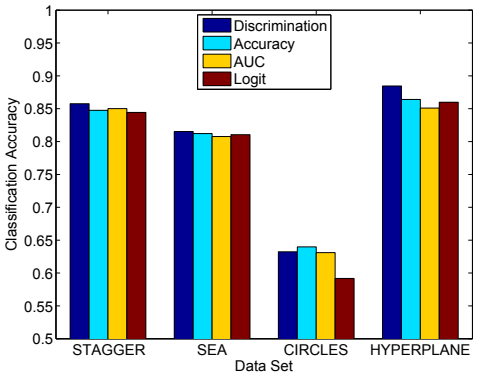


Figure 3: Comparison of rule qualities in model updating.

7.6 Comparison to Other Approaches

We now compare our approach against the well known Ensemble approach [12] and STEPD [8]. A brief description of the Ensemble approach is provided in Section 2. STEPD is a measure of detecting drifts based on the principle that if a drift is not occurring, then the accuracy on the most recent instances and the overall classification accuracy are not significantly different. A particular statistical test of equal proportions (STEPD) is utilized to determine if significant difference between the accuracies is observed. It is trivial to incorporate such a statistic into SRLF and use it on each rule to determine the occurrence of partial drifts.

Figure 4 shows the performance of SRLF with the χ^2 test for drift detection and rule discrimination for model updating and instance forgetting, against that of the STEPD and Ensemble approaches on all data sets, synthetic and real. STEPD on some data sets performs better than the Ensemble and on some performs worse. However, SRLF consistently outperforms both measures over all the data sets. This is most apparent from the CIRCLES data set where the concept changes very slowly, showing partial and gradual drifts. Based on the results on all the data sets, it can be concluded that SRLF is capable of producing high performance classification models in case of either global and sudden drift (as in STAGGER) or partial and gradual drift (as in SEA, CIRCLES and HYPERPLANE). Figure 4(c) is the average running time of the three models on each data set. Both STEPD and SRLF are more efficient than the Ensemble approach. SRLF is slightly slower than STEPD, but offers a significantly better classification result.

SRLF also offers better sensitivity compared to STEPD and Ensemble as shown in Table 5. All the three approaches

Table 4: The effect of the drop threshold on classification accuracy of rule quality based detection method.

	5%	15%	25%	35%	45%	55%	65%	75%	85%	95%
STAGGER	0.802	0.799	0.809	0.805	**0.815**	0.808	0.808	0.810	0.792	0.785
SEA	0.711	0.718	0.724	0.707	0.753	**0.730**	0.738	0.724	0.718	0.703
CIRCLES	0.568	0.550	0.562	0.562	0.563	0.556	0.573	0.584	0.605	**0.608**
HYPERPLANE	0.808	0.801	0.808	**0.809**	0.791	0.791	0.782	0.797	0.773	0.761

Table 5: Sensitivity of SRLF based, Ensemble, and STEPD.

	STAGGER			SEA			CIRCLES		
TPR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FPR	0.5	1.0	0.5	0.5	1.0	1.0	0.7	1.0	1.0
LAG	0	0	0	0	0	0	0	0	0
	SRLF	Ensemble	STEPD	SRLF	Ensemble	STEPD	SRLF	Ensemble	STEPD

have the perfect TPR and lag on the tested data sets. But in terms of FPR, STEPD behaves similar to the rule quality based drift detection methods and is over sensitive. The Ensemble is the most sensitive method with FPR=1 on all the data sets because it learns a classifier on each new chunk of data. SRLF offers the lowest FPR. It is likely however to overreact to the incoming stream. Nonetheless, it performs better than STEPD and Ensemble in terms of FPR.

7.7 Comparison of Global and Partial Detection Methods

In this section we compare the sensitivity of global and partial methods of detecting drifts to determine if focusing on partial drifts is more beneficial than global drift detection. We use our SRLF framework with the χ^2 test, entropy, and rule accuracy measures for detecting partial drifts, and compare them to approaches that use the same measures for global drift detection. To this end, we introduce two global drift detection methods that use accuracy and the χ^2 test. The global accuracy method monitors the model classification accuracy from one chunk of data to the next and learns a new classification model from the most recent instances whenever the drop in classification accuracy is below a user defined threshold. The global χ^2 method is essentially the same, but instead of model accuracy, it monitors the change in the class distribution of entire instance space from one window to the next using the χ^2 test. A statistically significant drop suggests occurrence of a drift and the need for updating the current model. An entropy based global detection method is introduced in [11]. We further compare these approaches against the KL-divergence [5] global detection method. Brief descriptions of the global entropy and KL-divergence methods are provided in Section 2. Both approaches were designed only for detection of drifts and do not involve learning and maintaining a classification model. The result is provided in Table 6, where G in front of a method name denotes the *global* variant.

As shown in the table, the global entropy-based and KL-based methods are highly insensitive and do not detect any drift on these datasets (with TPR=FPR=0)[1]. On the other hand, the local entropy-based method has higher TPR and FPR than its global variant on all the datasets. Looking at the results for χ^2 and accuracy based methods, the local methods does not miss any drift (TPR=1). But the global ones miss most of the drifts on CIRCLES and the accuracy-based method misses some drifts on SEA as well. On the other hand, the global accuracy-based method has lower FPR than its local variant. The global χ^2 based method has lower FPR on STAGGER and SEA but higher FPR on CIRCLES than its local variant. Since CIRCLES exhibits gradual and partial drifts, we can say that the local χ^2 method is more successful on datasets with gradual and partial drifts than it is on datasets with global and sudden drifts. Among the three local methods evaluated here, the χ^2 based method clearly outperforms others.

[1]These results do not suggest the ineffectiveness of these approaches in other scenarios.

7.8 Analysis of Parameters

Figure 5 examines the effect of a few parameters on performance. The HYPERPLANE data are used in the experiments shown in this figure. Figure 5(a) shows the effect of the dimensionality of the drift. As the percentage of the attributes involved in the drift increases, the classification becomes a harder problem, and accuracy suffers. Figure 5(b) shows the magnitude of the drift which also negatively influences the model performance. Figure 5(c) shows the expected improvement of model performance with increasing number of instances per chunk. Finally, Table 7 shows the changes in accuracy with respect to changes in the rule coverage. For STAGGER, increasing the coverage requirement can be beneficial since rules learned from STAGGER generally have good coverage. On the other data sets, rules often do not have high coverage. As a result, increasing the coverage does not affect the performance at all since none of the rules pass the coverage threshold.

8. CONCLUSION

We have proposed a novel approach to detecting concept drifts using the χ^2 test alongside rule quality measures to responsively update the model to reflect the most current trends. The focus of this detection mechanism was on sensitivity to partial drifts since they are the most common form of drifts. We also proposed an instance forgetting strategy that facilitates maintenance of instances that are considered recent and also relevant to the current concept in order to reduce the variance of data distribution which allows for construction of more stable and accurate models. We compared our approach to a variety of established approaches from the literature on both synthetic and real data sets. We showed that our approach achieves better classification accuracy without sacrificing efficiency. We also examined the sensitivity of each model through computation of TPR, FPR and lag and showed that our approach can detect both local and global drifts effectively and is particularly successful in detecting gradual, partial drifts.

9. ACKNOWLEDGEMENTS

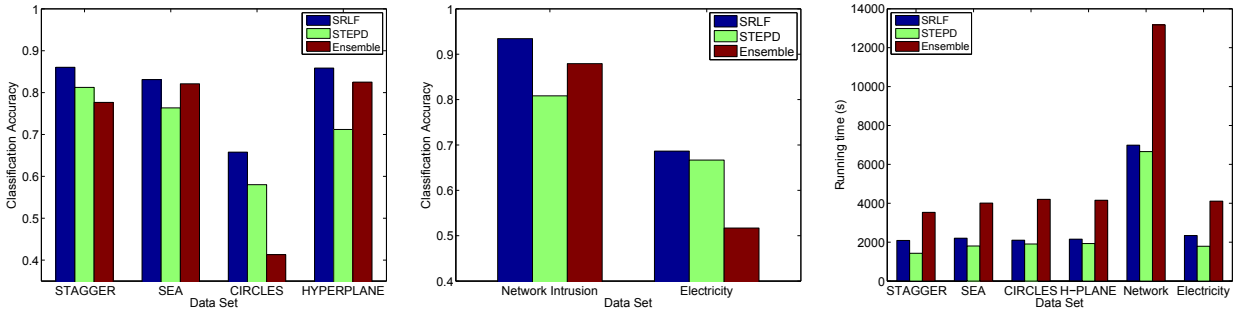
We would like to thank Prof. Ke Yi for kindly providing the code for the KL-divergence based drift detection method.

Table 6: Sensitivity of partial and global detection methods.

	STAGGER							SEA							CIRCLES						
TPR	1.0	1.0	1.0	1.0	0.5	0	0	1.0	1.0	1.0	0.7	0.5	0	0	1.0	0.2	1.0	0.2	0.5	0	0
FPR	0.5	0.2	0.5	0.5	0.5	0	0	0.5	0.25	1.0	0.5	0.5	0	0	0.7	0.8	1.0	0.8	0.5	0	0
LAG	0	0	0	0	0	∞	∞	0	0	0	0.25	0	∞	∞	0	1	0	0	0	∞	∞
	χ^2	G.χ^2	Ac.	G.Ac.	Ent.	G.Ent.	KL	χ^2	G.χ^2	Ac.	G.Ac.	Ent.	G.Ent.	KL	χ^2	G.χ^2	Ac.	G.Ac.	Ent.	G.Ent.	KL

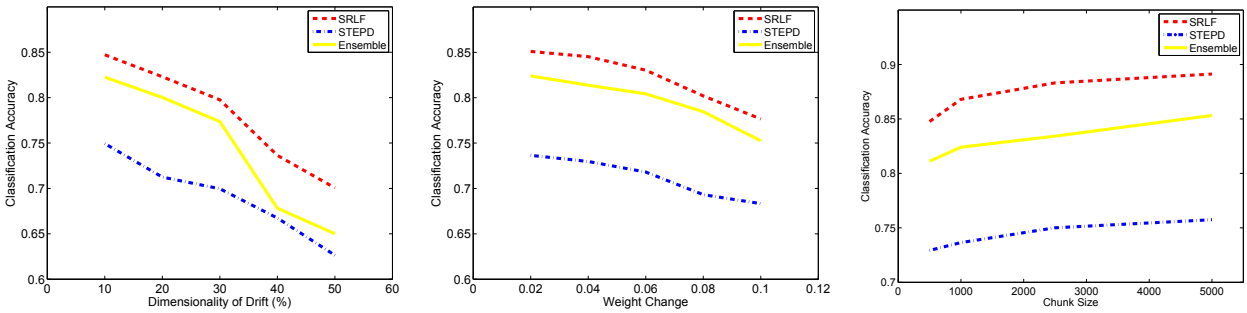
Table 7: The effect of rule coverage on classification accuracy of SRLF.

	1%	5%	10%	15%	20%	25%
STAGGER	0.844	0.857	0.857	0.865	0.865	0.865
SEA	0.802	0.815	0.815	0.815	0.815	0.815
CIRCLES	0.581	0.632	0.632	0.632	0.632	0.632
HYPERPLANE	0.833	0.884	0.884	0.884	0.884	0.884



(a) Accuracy on synthetic data sets. (b) Accuracy on real data sets. (c) The average running time.

Figure 4: Comparison of performance of the χ^2 based, Ensemble, and STEPD and approaches.



(a) Dimensionality of Drift. (b) Magnitude of Drift. (c) Chunk Size.

Figure 5: The effect of parameters on performance.

10. REFERENCES

[1] Concept drift. `http://en.wikipedia.org/wiki/Concept_drift`.

[2] C. Aggarwal, J. Han, J. Wang, and P. Yu. On demand classification of data streams. In *In Proc. of the 10th ACM SIGKDD*, pages 503–508. ACM, 2004.

[3] A. An and N. Cercone. ELEM2: A learning system for more accurate classifications. *Lecture Notes in Computer Science*, 1418:426–441, 1998.

[4] A. An and N. Cercone. Rule quality measures for rule induction systems: Description and evaluation. *Computational Intelligence*, 17(3):409–424, 2001.

[5] T. Dasu, S. Krishnan, S. Venkatasubramanian, and K. Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *38th Symposium on the Interface of Statistics, Computing Science, and Applications*. Citeseer.

[6] J. Gama, P. Medas, G. Castillo, and P. Rodrigues. Learning with drift detection. *Advances in Artificial Intelligence–SBIA 2004*, pages 286–295, 2004.

[7] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of the 7th ACM SIGKDD*, pages 97–106. NY, USA, 2001.

[8] K. Nishida and K. Yamauchi. Detecting concept drift using statistical testing. In *Discovery Science*, pages 264–269. Springer, 2007.

[9] W. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *In Proc. of the 7th ACM SIGKDD*. NY, USA, 2001.

[10] A. Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 2004.

[11] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *ICDM'06.*, pages 1113–1118, 2006.

[12] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *ACM SIGKDD*, pages 226–235, 2003.

[13] L. Wang and A. An. Fast Counting with AV-Space for Efficient Rule Induction. *Proc. of the SDM*, 2007.

[14] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[15] Y. Yang, X. Wu, and X. Zhu. Combining proactive and reactive predictions for data streams. *In Proc. of the 11th ACM SIGKDD*, page 715, 2005.