# Mining in Anticipation for Concept Change: Proactive-Reactive Prediction in Data Streams *

Ying Yang[1], Xindong Wu[2], and Xingquan Zhu[2]

[1] School of Computer Science and Software Engineering,
Monash University, Melbourne, VIC 3800, Australia,
yyang@csse.monash.edu.au
[2] Department of Computer Science,
University of Vermont, Burlington, VT 05405, USA
{xwu,xqzhu}@cs.uvm.edu

**Abstract.** Prediction in streaming data is an important activity in the modern society. Two major challenges posed by data streams are (1) the data may grow without limit so that it is difficult to retain a long history of raw data; and (2) the underlying concept of the data may change over time. The novelties of this paper are in four folds. First, it uses a measure of *conceptual equivalence* to organize the data history into a history of concepts. This contrasts to the common practice that only keeps recent raw data. The concept history is compact while still retains essential information for learning. Second, it learns concept-transition patterns from the concept history and anticipates what the concept will be in the case of a concept change. It then *proactively* prepares a prediction model for the future change. This contrasts to the conventional methodology that passively waits until the change happens. Third, it incorporates *proactive* and *reactive* predictions. If the anticipation turns out to be correct, a proper prediction model can be launched instantly upon the concept change. If not, it promptly resorts to a *reactive* mode: adapting a prediction model to the new data. Finally, an efficient and effective system *RePro* is proposed to implement these new ideas. It carries out prediction at two levels, a *general* level of predicting each oncoming concept and a *specific* level of predicting each instance's class. Experiments are conducted to compare RePro with representative existing prediction methods on various benchmark data sets that represent diversified scenarios of concept change. Empirical evidence offers inspiring insights and demonstrates the proposed methodology is an advisable solution to prediction in data streams.

**Key Words:** Data Stream; Concept Change; Classification; Proactive Learning; Reactive Learning; Conceptual Equivalence.

* A preliminary and shorter version of this paper has been published in the Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2005), pages 710-715.

## 1 Introduction

Prediction in data streams is important. Streaming data are very common in science and commerce nowadays such as meteorology statistics, Internet event logs and stock market transactions. Streaming data have two special characteristics. First, the data items arrive in sequence along the time line and their amount may grow without limit. Second, the data's underlying concept may change over time. Prediction is critical in many aspects of the modern society, where people need to know what will happen in the (sometimes very near) future. For example, various meteorologic measures, such as heat, humidity and wind speed, are indicative of the weather condition. A model can be learned from previous observations to describe the relationship between the measures and the weather. When new measures are taken, this model can be used to predict what the weather will be. Particularly this paper is set in the context of classification learning.

Prediction in data streams is, however, not a trivial task. The special characteristics of streaming data have brought new challenges to the prediction task. First, the prohibitive data amount makes it infeasible to retain all the historical data. This limited access to the past experience may obstruct proper prediction that needs to collect evidence from the history. Hence it requires a cleverer way to organize the history. Second, the underlying concept change demands updating the prediction model accordingly. For example, the weather pattern may change across seasons. A prediction model appropriate for spring may not necessarily suit summer. Hence, the prediction process is complicated by the need to predict, not only the outcome of a particular observation, but also the oncoming concept in case of a concept change.

Substantial progress has been made on prediction in data streams [1–8]. With all due respect to previous achievements, this paper suggests that some problems still remain open. First, the history of data streams is not well organized nor made good use of. Two typical methodologies exist. One is to keep a recent history of raw instances since the whole history is too much to retain. As will be explained later, this methodology may be applicable to *concept drift* but not proper for *concept shift*. Another methodology, upon detecting a concept change, discards the history corresponding to the outdated concept and accumulates instances to learn the new concept from scratch. As will be demonstrated later, this methodology ignores experience and incurs waste when the history sometimes repeats itself. A second open problem is that existing approaches are mainly interested in predicting the class of each specific instance. No significant effort has been devoted to foreseeing a bigger picture, that is, what the new concept will be if a concept change takes place. This prediction, if possible, is at a more general level and is *proactive*. As human beings have learned to foresee a winter after an autumn and prepare for the cold weather in advance, proactive learning has been shown to be useful in practice for many aspects of life. It is desirable to bestow the same merit on stream data mining systems. By doing so, it helps prepare for the future concept change and can instantly launch a new prediction strategy upon detecting such a change.

This paper proposes elegant and effective solutions to the above important but difficult problems. In particular, the new solutions organize the history of raw data into a history of concepts using a conceptual equivalence measurement, learn transition patterns among concepts, and accomplish proactive prediction in addition to reactive prediction. All those tasks are carried out in an online fashion along the journey of concept change. To the best of the authors' knowledge, this paper is the first to advocate and address 'proactive learning' in mining concept-change data streams. Section 2 introduces some background knowledge. It defines terms used throughout the paper. It explains and compares different modes of concept change. It also gives taxonomies of prediction approaches for streaming data. Section 3 proposes a mechanism to dynamically build a history along the journey of concept change by identifying new concepts as well as re-appearing historical ones. Section 4 introduces a system RePro that learns a concept transition matrix from the concept history, and predicts for streaming data in a proactive-reactive manner. Section 5 differentiates this paper from related work. Section 6 presents empirical evaluations and discussions. Section 7 gives concluding remarks.

## 2 Background knowledge

This section prepares background knowledge for a better understanding of this paper.

### 2.1 Terminology

This paper deals with prediction in the context of classification learning. A **data stream** is a sequence of **instances**. Each instance is a vector of **attribute** values. Each instance has a **class** label. If its class is known, an instance is a **labeled** instance. Otherwise it is an **unlabeled** instance. Predicting for an instance is to decide the class of an unlabeled instance by evidence from labeled instances.

The term **concept** is more subjective than objective. Facing the same data, different people may see different concepts according to different perspectives or interests. Particularly in this paper, a concept is represented by the learning results of a classification algorithm, such as a classification rule set or a Bayesian probability table.

### 2.2 Modes of concept change

The following different modes of change have been identified in the literature[3]:

$$\text{modes} \begin{cases} \text{concept change} \begin{cases} \text{concept drift} \\ \text{concept shift} \end{cases} . \\ \text{sampling change} \end{cases}$$

[3] Sometimes there are conflicts in the literature when describing these modes. For example, the concept shift in some papers means the concept drift in other papers. The definitions here are cleared up to the best of the authors' understanding.

**Concept change** refers to the change of the underlying concept over time. **Concept drift** describes a *gradual* change of the concept [8–10]. For example, a slow wearing piece of factory equipment might cause a gradual change in the quality of output parts. **Concept shift** [9, 11] happens when a change between two concepts is more abrupt. For example, in the history of US politics, many aspects of the national policies change whenever a different party takes the government. There are not necessarily gradual transitions between a republican government and a democrat one. **Sampling change**, also known as sampling shift [12] or virtual concept drift [8], refers to the change in the data distribution. Even if the concept remains the same, this change may often lead to revising the current model as the model's error rate may no longer be acceptable with the new data distribution.

Stanley [9] has suggested that from the practical point of view, it is not essential to differentiate between concept change and sampling change since the current model needs to be changed in both cases.

This paper further suggests that it is important to distinguish between concept drift and concept shift. The difference between concept drift and concept shift resembles the difference between a numeric variable and a nominal variable. For example, it is meaningful to say that the value 5.3 is closer to 5.4 than to 5.5. It is not meaningful in the same way to say that a piano is more similar to a violin than to a flute. Analogously, one may assume that the current concept is always the most similar to the oncoming concept during concept drift because of its gradual change [7], whereas this assumption can often be violated in concept shift. Both cases of concept drift and concept shift abound in the real world. Since they have different characteristics, they may require different optimal prediction strategies.

### 2.3 Taxonomy of prediction approaches

Many prediction approaches in streaming data have been proposed. Integrating various previous proposals and its own new perspectives, this paper presents a comprehensive set of taxonomies, each of which emphasizes a different aspect of the distinctions among prediction approaches.

**Trigger-sensitive** *vs.* **Trigger-insensitive**. Trigger-sensitive approaches detect triggers, instances across which the underlying concept changes. Once a trigger is detected, a new model is constructed for data coming after the trigger. As a result, the data are partitioned into segments according to triggers, each having a different underlying concept. Hence these approaches are also called segmentation algorithms [4]. Trigger-insensitive approaches do not explicitly detect triggers. Instead, they continuously adapt the current model to newly coming instances [3].

**Incremental** *vs.* **Batch**. Incremental approaches process coming instances one by one. Batch approaches exam a batch of instances at once [7, 10, 13].

**Historical** *vs.* **Contemporary**. After a trigger is detected, historical approaches consult the history to construct a new model while contemporary approaches only resort to data in hand that have just triggered the concept change.

**Proactive** *vs.* **Reactive**. Proactive approaches foresee what the forthcoming concept is given the current concept. This predicted concept will take effect once a concept change is detected. Reactive approaches do not predict what concept is coming. A new prediction model is constructed upon a trigger is detected.

## 3 Building concept history

A mechanism is proposed here to build a concept history along the journey of concept change by identifying new concepts as well as re-appearing ones. This history serves three purposes. First, it retains essential information of the past data. It is important to record the history especially when it may repeat itself. However, it is very expensive, if possible at all, to keep all the streaming data whose volume is prohibitive. In comparison, a concept is compact such as a bunch of abstract rules. Keeping a long history of concepts is much more tractable. Second, it stores previous concepts so that they can be reused in the future if needed. This helps reduce the prediction time than learning from scratch each time when the concept changes in streaming data. Third, the possible associations among different concepts can be learned according to the history. This type of learning has taken place and is very valuable through human beings' history. For example, people have learned over time that there exists certain transitions among seasons, such as winter coming after autumn. As a result, people have been preparing for winter during autumn, which is of great help in many aspects of life. The following components are key to building a concept history.

### 3.1 A classification algorithm

This algorithm is used to abstract a concept from the raw data. A user may choose whatever classification algorithm of his/her interest, or choose one that appears to be good at learning the current data. The proposed system accepts diversified formats of learned concepts, such as decision rules, decision trees or even probability tables from the naive-Bayes learning. In this particular paper, C4.5rules [14] is employed since it is commonly used and can achieve a reasonable classification accuracy in general.

### 3.2 A trigger detection algorithm

This algorithm finds instances, across which the underlying concept has changed and the prediction model should be modified. It is especially important when concept shift happens, where the change may not be gradual and hence the immediate previous concept may not be the best simulator of the new concept. For example, suppose a democrat government succeeds a republican government. To predict what national policies will take place, it is better to check what happened under previous democrat governments in history, rather than check what happened under the last republican one although it was the most recent.

Keogh and Tsymbal et al. [4, 10] have reviewed numerous trigger detection algorithms. A *sliding-window* methodology is used here. Two important parameters are the *window size* and the *error threshold*. The current prediction model predicts the class of coming instances one by one. The beginning of the window is always a misclassified instance. Whenever the window is full and its error rate exceeds the error threshold, the beginning instance is taken as a trigger; otherwise, the beginning of the window is slid to the next misclassified instance (if there is any) and the previous instances are dropped from the window.

### 3.3 A measure of conceptual equivalence

This measure checks whether a concept is brand new or reappearing. Given current data and the corresponding learned concept $V$, this measure calculates the degree of equivalence between this concept and each distinct historical concept $T$, as in Table 1. If the measure between $V$ and $T$ is above a user-defined threshold, the system deems that $V$ is a reappearing $T$ and substitutes $T$ for $V$ when putting it into the concept history. Otherwise, $V$ is new and itself is put into the history.

**Input**: concept $V$ newly learned from current data $D$, historical concept $T$
**Output**: $ce \in [-1, 1]$, the bigger the value, the higher the degree of conceptual equivalence between $V$ and $T$
**Begin**
$ce = 0$;
FOREACH instance $I \in D$
  $result1$ = classify $I$ by $V$;
  $result2$ = classify $I$ by $T$;
  IF ($result1$ != $result2$) $score$=-1;
  IF ($result1$ == $result2$)
    IF ($result1$ == nomatch) $score$=0;
    ELSE $score$=1;
  $ce$ += $score$;
$ce = ce$ / $|D|$;
return ($ce$);
**End**

**Table 1.** Measuring conceptual equivalence

A few insights are worth mentioning in this algorithm. First, the result of classifying an instance by a concept can be 'approve', 'nomatch' or 'conflict_*class*' where *class* is a concept's verdict on an instance. Conflict_$X$ is different from conflict_$Y$. For example, if $V$ misclassifies $I$ into class $X$ while $T$ misclassifies $I$ into class $Y$, the score is -1. If both misclassify $I$ into $X$, the score is +1. If neither $V$ nor $T$ match $I$, the score is 0 since there is no strong evidence to judge whether $I$ stands for a discrepancy or an equivalence between $V$ and $T$.

Second, the measure circumvents syntactically comparing two rule sets. In streaming data, instances from the same concept may not necessarily duplicate at different times, especially when attributes involve continuous values. As a result, two rule sets for a single concept can differ in their appearances. On top of that, some popular online learning algorithms, such as naive-Bayes, do not produce explicit rules at all, where direct rule comparisons are hence inapplicable.

Third, the measure is different from the conceptual equivalence measure used in the previous work [15], which judges by the classification accuracies of $V$ and $T$ on $D$ and is not appropriate here. For example, the accuracy and the conceptual equivalence degree are not necessarily positively correlated. Although $V$ and $T$ classify $D$ with low accuracies, their equivalence degree can be very high if they agree on classifying many instances even when their classifications are both wrong. Also, accuracy equivalence does not indicate conceptual equivalence. Suppose both $V$ and $T$ classify $D$ with accuracy 80%. However their individual 20% misclassifications may stem from different portions of $D$. Hence, they are actually quite different.

### 3.4 A stable learning size

This is a parameter that specifies the number of instances above which the learned concept can be deemed stable. In order to be sensitive to concept change, the window size of trigger detection is normally small. Sometimes this window of instances are *sufficient* to indicate that the current concept is not adequate any more, but *insufficient* to induce what the adequate concept should be. If one has to learn a new concept out of these few instances, the learned concept is not stable. For example, it is of high classification variance by overfitting a small portion of instances. Hence the concept should be re-learned once a stable learning size of instances are accumulated.

In practice, the system keeps two concept histories: a stable $\overline{H}$ retaining stable concepts and triggers; and a temporary $H$ retaining possibly unstable concepts and triggers. Assume the stable learning size is $s$ and the instance index for the last stable trigger is $j$. Each concept $C_t$ learned between instance $j$ and $j+s$ is stored in $H$. When $s$ instances' classes since $j$ are known, a single concept $C_k$ is learned across $[j,j+s)$. If the classification accuracy of $C_k$ is no less than the average classification accuracy of $C_t$s on these $s$ instances, $C_k$ is deemed as a stable concept and is stored into $\overline{H}$; and $H$ is updated by $\overline{H}$. Otherwise, $C_t$s are deemed as a series of stable concepts and $\overline{H}$ is updated by $H$. This strategy is instructed by the theory of Occam's razor that one should not increase, beyond what is necessary, the number of entities required to explain anything. Although $H$ may be temporarily used to predict each instance's class, $\overline{H}$ is always used to learn transition matrices and to predict the oncoming concept.

### 3.5 The building process

Integrating all the above key components, the process of building a concept history is illustrated in Table 2. To more clearly explain the process, assume

that the window size is 10, the stable learning size is 30 and the error threshold is 55%. A ♠ represents an instance where a stable trigger is detected. A ♣ represents an instance where a temporary trigger is detected. A $\surd$ represents a correctly classified instance. A $\times$ represents a misclassified instance. Only instances coming after the last stable trigger and up to now are retained. Other historical instances are not essential to keep.

## 4 Choosing prediction models

As mentioned in Stage 3 of Table 2, when a new trigger is detected, it indicates that the current prediction model is no longer appropriate. A different model needs to be chosen to classify oncoming instances. The following information is available to make this choice: (1) a history of concepts up to this trigger; and (2) a window size of labeled instances, which contributed to detect this new trigger. Three different approaches to choosing a prediction model are studied here.

### 4.1 Proactive

According to Section 2.3, proactive modeling is trigger-sensitive, incremental, historical and proactive. A proactive approach predicts the oncoming concept given the current concept(s) by evidence from the concept history. The choice can be made ahead of a trigger detection and is independent of the trigger window. Once a new trigger is detected that indicates the concept has changed, the predicted concept immediately takes over the classification task.

In the proactive style, the history of concepts is treated like a Markov Chain [16]. Markov chain is a commonly used model for web access path prediction in information retrieval. It involves a sequence of states where the probability of the current state depends on the past state(s). If the probability only depends on the immediate past state, it is a first-order Markov chain. Otherwise it is a higher-order Markov chain.

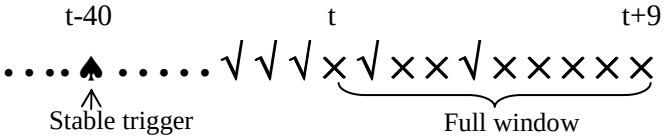
In the context of proactive modeling, each distinct concept is a state. For example, the weather patterns normally change following the order of spring, summer, autumn and winter. Besides, due to occasional special climate conditions, abnormal concepts can take place from time to time, such as the consecutive Hurricanes Charley of Year 2004 and Katrina of Year 2005 in south America. Suppose the history of concepts is: *spring, summer, autumn, winter, spring, summer, hurricane, autumn, winter, spring, flood, summer, autumn, winter, spring, summer, autumn, winter, spring, summer, hurricane, autumn, etc.* A transition matrix can be constructed from the concept history and dynamically updated upon each future occurrence of concept change. Table 3 shows the current status of a first-order transition matrix[4].

Suppose that the current concept is autumn. According to the transition matrix, in history the most frequent concept after autumn is winter. Hence,

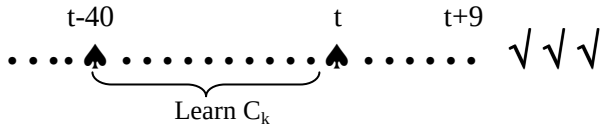
[4] The value in each cell can be frequency as well as probability. The latter can be approximated from the former.

**Stage 0** Initially, one can randomly guess each instance's class. When there are a stable-learning-size of instances with known classes, the first concept can be learned and stored in the stable history.

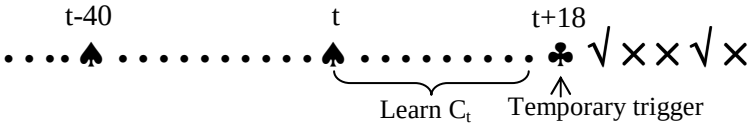
**Stage 1** Starting from t, errors are witnessed. Up to t+9, the window is full and the error rate is 80% which is above the error threshold. Hence, t is detected as a new trigger.




**Stage 2** Since there are no less than 30 instances between t-40 (last stable trigger) and t, the concept $C_k$ learned from [t-40, t) is stable. Use conceptual equivalence (as in Table 1) to check whether $C_k$ is a historical concept or a new one. Store $C_k$ in the stable history. Update the contemporary history by the stable one.




**Stage 3** A new prediction model is chosen for instances coming after t+9. Assume a new trigger is detected at t+18. A concept $C_t$ learned over [t, t+18) can be unstable since it is learned from too few instances. Store it only into the temporary concept history.




**Stage 4** When time arrives at t+29, there have accumulated 30 instances. Learn a concept $C_{k+1}$ from [t, t+29]. If $C_{k+1}$'s accuracy is competitive to temporary $C_t$'s, store $C_{k+1}$ into the stable history and update the temporary history by the stable one. Otherwise update the stable history by the temporary one.

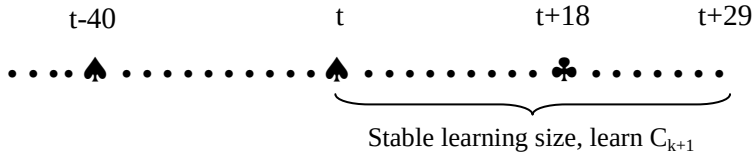



**Table 2.** The process of building concept history

| Current State | Next State | | | | | | |
|---|---|---|---|---|---|---|---|
| | spring | summer | autumn | winter | hurricane | flood | ... |
| spring | 0 | 4 | 0 | 0 | 0 | 1 | |
| summer | 0 | 0 | 3 | 0 | 2 | 0 | |
| autumn | 0 | 0 | 0 | 4 | 0 | 0 | |
| winter | 4 | 0 | 0 | 0 | 0 | 0 | |
| hurricane | 0 | 0 | 2 | 0 | 0 | 0 | |
| flood | 0 | 1 | 0 | 0 | 0 | 0 | |
| ... | | | | | | | |

**Table 3.** Transition Matrix

the system will predict that if there is a concept change from autumn, it is very likely to be winter. Accordingly, once a new trigger is detected, the concept winter is used to classify oncoming instances. The advantages are that the future model can be chosen in advance; and the reaction to concept change is fast. A problem happens when there is no state with a dominant probability given the current state. For example, if the current state is 'summer', the frequency of autumn being the next one is 3 while the frequency of hurricane is 2 in history. In this case, the first-order transition matrix does not offer enough information to distinguish between autumn and hurricane. One may turn to a second-order transition matrix, that is, consult two states back into history to predict the next state. If the tie still can not be broken, one may use a third-order transition matrix and so on. However, the higher order the matrix to maintain, the more expensive the system becomes. Or, one can incorporate a reactive approach into the proactive approach to break ties, which is the topic of the following section.

It is worth emphasizing here that the transition matrix is dynamically updated. Each time when a concept change happens, the statistics among concept transitions will be changed accordingly. Each time when a new concept appears, a new column and a new row are added into the matrix corresponding to it. Proactive predictions are always conducted using the updated transition matrix.

### 4.2 Reactive

A reactive approach chooses a model according to the trigger instances.

**Contemporary** According to Section 2.3, contemporary-reactive modeling is trigger-sensitive, incremental, contemporary and reactive. Upon detecting a new trigger, contemporary-reactive prediction does not consult the concept history. To train a prediction model, it simply uses the window of labeled instances upon the new trigger. Then it uses this model to classify oncoming instances. Although straightforward, this approach risks high classification variance especially when the sliding window is small. For example, C4.5rules can almost always form a set of rules with a high classification accuracy across a window of instances. This prediction model, however, is very likely to wrongly classify oncoming instances

since it poorly generalizes to the true underlying concept. As a result, another new trigger is soon detected even when the data are from the same concept. Nonetheless, it can be a measure of last resort if the concept is brand new and has no history to learn from.

**Historical** According to Section 2.3, historical-reactive modeling is trigger-sensitive, incremental, historical and reactive. Upon detecting a new trigger, historical-reactive prediction retrieves a concept from the history that is most appropriate for the trigger instances. It tests each distinct historical concept's classification error across the trigger window. The one with the lowest error is chosen as the new prediction model. One merit of consulting the concept history is that each concept accepted by the history is a stable classifier. Hence it can avoid the problem taking place in the contemporary-reactive modeling. However, there are still potential disadvantages. One problem happens when this new concept is very different from every existing concept. Consequently, the history offers a low classification accuracy on the window. Another potential concern is the efficiency issue. If there are many distinct concepts in history, it may take a while to test every single one across the window.

### 4.3 RePro, a coalition of strength

As addressed above, reactive and proactive modeling each excels in different scenarios. A system RePro (reactive upon proactive), as in Table 4, incorporates both and uses one's strength to offset the other's weakness.

As a result, if the proactive component of RePro foresees multiple probable concepts given the current concept, one can use the historical-reactive component as a tie breaker. The other way around, if there are many historical concepts, the proactive component can offer a short list to the historical-reactive component and speed up the process. If both the proactive and historical-reactive modeling incur low accuracy in the new trigger window, it indicates that the new concept is very different from historical ones. Hence, a contemporary-reactive classifier can help to cope with the emergency. Nonetheless, as explained before, in order to avoid the high classification variance problem, when a stable-learning-size of labeled instances are accumulated, one should update the contemporary-reactive concepts by a stable concept.

## 5 Related work

Many approaches have been published that predict in concept-changing scenarios. Three representative mechanisms are discussed here. More comprehensive reviews can be found in various informative survey papers [4, 10].

The FLORA system [8] is related to RePro in the sense that it stores old concepts and reuses them if appropriate. However, it is oriented to data of small sizes instead of streaming data [3]. It represents concepts by conjunctions of

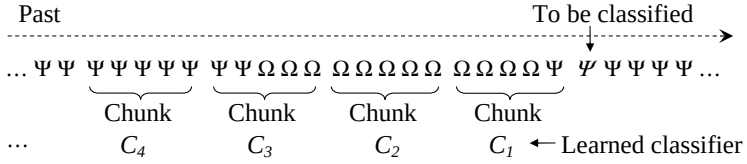
**Input**: a list of distinct historical concepts $C\_DIS$ distinguished by the conceptual equivalence measure as in Table 1, a concept history $C\_HIS$ learned as in Table 2, a transition matrix $C\_TRA$ learned from $C\_HIS$ as in Table 3, a window of trigger instances $I\_WIN$, a probability threshold $threshold_{prob}$, a classification accuracy threshold $threshold_{accu}$
**Output**: a concept (prediction model) $c_{next}$ for oncoming instances
**Begin**
$c_{last}$ = the last stable concept in $C\_HIS$;
// Proactive
$c_{next}$(s) = concept(s) whose probability given $c_{last}$ is bigger than $threshold_{prob}$, according to $C\_TRA$;
IF multiple $c_{next}$s exist
  // Break tie by historical-reactive
  FOREACH $c_{next}$
    calculate its accuracy on $I\_WIN$;
  $c_{next}$ = the one acquiring the highest accuracy;
// Historical-reactive
IF no $c_{next}$ exists
  FOREACH concept $c_{historical} \in C\_DIS$
    calculate its accuracy on $I\_WIN$;
  $c_{next}$ = the one acquiring highest accuracy;
// Contemporary-reactive
IF accuracy of $c_{next}$ on $I\_WIN$ is less than $threshold_{accu}$
  $c_{next}$ = concept learned from $I\_WIN$;
return ($c_{next}$);
**End**

**Table 4.** RePro: reactive+proactive

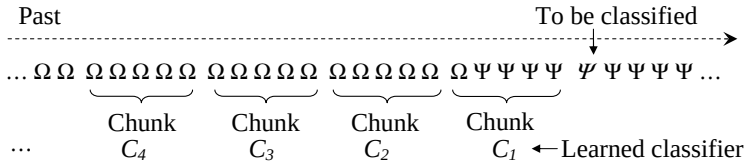
attribute values and measures the conceptual equivalence by syntactical comparisons. This is less applicable in streaming data, as will be demonstrated in Section 6.3. The concepts are stored only for a reactive purpose. FLORA does not explore their associations and hence can not be proactive.

The weighed classifier ensemble (WCE) approach [7] represents a big family of algorithms that ensemble learners for prediction. In this work, it is proved that a carefully weighed classifier ensemble built on a set of data partitions $S_1, S_2, \cdots, S_n$ is more accurate than a single classifier built on $S_1 \cup S_2 \cup S_n$. To classify a coming instance, WCE divides its previous data into sequential chunks of fixed size, builds a classifier from each chunk, and composes a classifier ensemble where each classifier is weighed proportional to its classification accuracy on the chunk most recent to the instance to be classified. WCE *relates* to RePro in the sense that it uses a history of concepts (classifiers). However, the quality of this history is controlled by an arbitrary chunk size $k$. There is no trigger detection nor conceptual equivalence. As a result, sub-optimal prediction accuracy can be witnessed in the following situations, which can be more striking when concept shifts than when concept drifts.

1. In the most recent data chunk $C_1$ that is the gauge to weigh classifiers in the ensemble, the majority of the instances are from the previous concept instead of the new one. As illustrated below, an instance of concept $\Psi$ is to be classified. Because the majority of instances in $C_1$ are of concept $\Omega$, classifiers suitable for $\Omega$ (such as $C_1$ and $C_2$) will be ironically given higher weights than those suitable for $\Psi$ (such as $C_4$).




2. WCE needs to update data chunks for learning classifiers upon receiving new labeled instances. Since it has to retain raw data, WCE can only hold a relatively recent history in the context of streaming data whose volume is otherwise prohibitive. If the previous concept dominates the recent history, consulting this history does not necessarily help prediction. As illustrated below, when classifying an instance of concept $\Psi$, most classifiers are still indulged in the previous concept $\Omega$. As a result, even when the most recent data chunk favors instances of $\Psi$, its correct weighing can be overshadowed by the large number of improper classifiers (a single high-weight $C_1$ against many low-weight $C_2, C_3$ and $C_4$).




The concept-adapting very fast decision tree (CVFDT) [3] is one of the best-known systems that achieves efficient classification in streaming data. It represents a popular methodology that continuously adapts the prediction model to coming instances. It starts with a single leaf and starts collecting labeled instances from a data stream. When it knows enough data so that it knows with high confidence which attribute is the best to partition the data with, it turns the leaf into an internal node, splits on that attribute and starts learning at the new leaves recursively. CVFDT maintains a window of training instances and keeps its learned tree up-to-date with this window by monitoring the quality of its old decisions as instances move into and out of the window. In particular, whenever a new instance comes, three things take place: (1) it is added to the statistics at all the nodes in the tree that it passes through, (2) the last instance

in the window is forgotten from every node where it had previously had an effect, and (3) the validity of all statistical tests are checked. CVFDT periodically scan the internal nodes of the tree looking for those where the chosen split attribute would no longer be selected. If this happens, CVFDT detects a concept change. It then starts growing an alternate tree in parallel which is rooted at the newly-invalidated node. When the alternate tree is more accurate on new data than the original one, the original is replaced by the alternate and freed. CVFDT *relates* to RePro in the sense that it is trigger-sensitive. However, upon detecting a concept change, CVFDT builds a new prediction model from scratch. To some extent, it resembles the contemporary-reactive modeling. In cases where history repeats itself, CVFDT can not take advantage of previous experience and hence may be less efficient. Between the gap where the old model is outdated and the new model has not matured, the prediction accuracy may suffer.

In summary, the taxonomy (according to Section 2.3) of each discussed approach is in Table 5.

| Method | Trigger | Learning | History | Action |
|---|---|---|---|---|
| RePro | sensitive | incremental | historical | proactive /reactive |
| FLORA | sensitive | incremental | historical | reactive |
| WCE | insensitive | batch | historical | reactive |
| CVFDT | sensitive | incremental | contemporary | reactive |

**Table 5.** Taxonomy of methods

In the following section of experimental evaluations, empirical comparisons will be conducted among RePro, WCE and CVFDT to verify the above understandings. FLORA is not involved since it is not oriented to handling streaming data's large amount.

## 6 Experiments

Experiments are conducted to evaluate the efficacy and efficiency of RePro for prediction in streaming data. In particular, three hypotheses are to be verified. First, RePro incorporates the strength of reactive and proactive prediction, and offers a rapport of high prediction accuracy and a low time consumption. Second, it is advisable to use conceptual equivalence when building a concept history, which improves prediction efficiency with little loss of prediction accuracy. Third, RePro is able to outperform existing popular prediction mechanisms for various types of concept changes.

### 6.1 Data

Many papers have been published dealing with streaming data. For experimental data, authors sometimes choose particular real-world data sets to which they

have private access. A potential problem is that these private data sets are seldom reusable by other researchers because of organizational policies. In their paper, Keogh and Kasetty have advocated using benchmark data sets that allow public access and allow fair comparisons among rival methods [4]. Kolter and Maloof agreed with this ideology by using only benchmark data sets in their experiments [5]. This paper also employs benchmark data sets.

The experiments involve both artificial data and real-world data. By using artificial data, one can access information such as what type of concept change is taking place and when. Hence alternative algorithms can be precisely tested to validate theoretical analyses offered by this paper. By using real-world data, one can verify RePro's strength in real-world scenarios. Please note that in real-world data streams, such as bank transaction data and stock market data, there is often a time lag between getting a new instance and knowing its true class. Accordingly the true concept change and the detection of the change might not be perfectly synchronized. Nonetheless, if each concept lasts for a while so that the class labels can be obtained and the concept can be learned before the next concept change, the time lag is tolerable.[5] Hence, as implied by many previous papers of data stream classification [3, 5–7], it is assumed in experiments here that the frequency of concept change is moderate and one can timely obtain the true classes of a majority of instances, which is consistent with many real-world cases.

**Artificial data** Three benchmark stream data sets are employed that cover all types of concept changes studied in Section 2.2: concept drift, concept shift and sampling change. When applicable, the instances are randomly generated taking time as seed. Hence, the reappearance of concepts does not necessarily mean the reappearance of instances, which better simulates the real world.

*Stagger* This data set can simulate the scenario of *concept shift* [5, 8–10]. Each instance consists of three attribute values: color $\in \{green, blue, red\}$, shape $\in \{triangle, circle, rectangle\}$, and size $\in \{small, medium, large\}$. There are three alternative underlying concepts, $\mathcal{A}$: if color $= red \wedge$ size $= small$, class= *positive*; otherwise, class= *negative*; $\mathcal{B}$: if color $= green \vee$ shape $= circle$, class= *positive*; otherwise, class= *negative*; and $\mathcal{C}$: if size $= medium \vee large$, class= *positive*; otherwise, class= *negative*. Particularly in this experiment, 500 instances are randomly generated according to each concept. Besides, one can simulate different real-world situations by controlling the transition among concepts from stochastic (say, $\mathcal{B}$ and $\mathcal{C}$ equally probably coming after $\mathcal{A}$) to deterministic (say, $\mathcal{B}$ always coming after $\mathcal{A}$). This is made possible by tuning the $z$ parameter of Zipfian distribution, which will be detailed in Appendix A to avoid distraction from the main text.

[5] If the concept changes so fast that the learning can not catch up with it, the prediction will be inordinate. This also applies to human learning.

*Hyperplane* This data set can simulate the scenario of *concept drift* by continuously moving a hyperplane [3, 5–7, 10]. A hyperplane in a $d$-dimensional space is denoted by equation: $\sum_{i=1}^{d} w_i x_i = w_0$, where each vector of variables $< x_1, ..., x_d >$ is a randomly generated instance and is uniformly distributed in the multidimensional space $[0, 1]^d$. Instances satisfying $\sum_{i=1}^{d} w_i x_i \geq w_0$ is labeled as positive, and otherwise negative. The value of each coefficient $w_i$ is continuously changed, as illustrated in Figure 1, so that the hyperplane is gradually drifting in the space. Besides, the value of $w_0$ is always set as $\frac{1}{2} \sum_{i=1}^{d} w_i$ so that roughly half of the instances are positive, and the other half are negative.

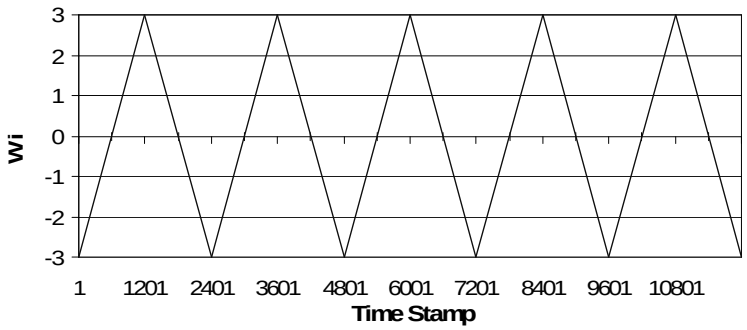



**Fig. 1.** To simulate concept drift, a hyperplane gradually moves by continuously changing each $w_i$. Particularly in this experiment, the changing range is [-3,+3] and the changing rate is 0.005 per instance.

*Network intrusion* This is another public accessible streaming data set and can simulate the scenario of *sampling change* [17]. This data set was used for the 3rd International Knowledge Discovery and Data Mining Tools Competition. It includes a wide variety of intrusions simulated in a military network environment. The task is to build a prediction model capable of distinguishing between normal connections (Class 1) and network intrusions (Classes 2,3,4,5). Different periods witness bursts of different intrusion classes, as in Figure 2. Assume all data are simultaneously available, a rule set can be learned that well classify each type of intrusion[6]. Hence one may think that there is only a single concept underlying the data. However, in the context of streaming data, a learner's observation is limited since instances come one by one. For example, the prediction model built during time [0, 786] will become incapable right afterwards.

**Real-world data** Classification data sets from the UCI machine learning repository [18] are used to produce data streams. Given a data set and one of its

[6] For example, C4.5rules [14] can achieve a 100% classification accuracy on the whole data set.


**Fig. 2.** Network intrusion is typical sampling change. Classes 1, 2... are not numeric but nominal.

attributes (including the class), if the attribute is nominal, each of its values is treated as a state in the Markov chain. A sequence of states is produced as explained in Appendix A with $z = 1$ of the Zipfian distribution. For each state, 500 instances possessing the same attribute value are randomly selected from the data set[7], and are then queued into a data stream. If the attribute is numeric, all instances are ascendingly sorted according to its value. The sorted instances are queued into a data stream. In theory, the length of the data stream can increase infinitely because one can produce a Markov chain of arbitrary length for a nominal attribute, as well as can repeat queueing an ordered set of instances for a numeric attribute. Experiments in this paper take 20,000 as the upper limit of the data stream length, which is a sufficient sample to show the learning trends of alternative algorithms.

A dozen of commonly-used UCI data sets are employed. For each data set, three data streams are produced, whose states correspond to a randomly-selected nominal attribute, a randomly-selected numeric attribute and the class respectively.[8] As a result, 36 real-world data streams are used for evaluating RePro. Each stream may or may not have (different types of) concept change. This poses a real-world challenge to prediction algorithms that in reality one seldom knows beforehand what type of concept change is happening. Hence a mechanism of choice should perform well in general, regardless of types of concept change.

### 6.2 Rival methods

As discussed in Section 5, two representative methods for prediction in streaming data are the weighed classifier ensemble (WCE) [7] and the concept-adapting

[7] If the attribute value has less than 500 instances, all instances will be sampled without replacement.

[8] If a data set has only nominal attributes, two nominal attributes will be selected. If a data set has only numeric attributes, two numeric attributes will be selected.

very fast decision tree (CVFDT) [3]. They are taken as straw men to empirically evaluate the proposed system RePro.

The original implementation of WCE specifies a parameter chunk size $s$. It does not update the ensemble until another $s$ instances have come and been labeled. Readers may be curious about the performance of WCE under different frequencies of updating the ensemble. Hence, a more dynamic version, dynamic_WCE (DWCE), is also implemented here. DWCE specifies two parameters,a chunk size $s$ and a buffer size $f$ (normally $f \leq s$). Instead of waiting for a full chunk, once $f$ new instances are labeled, DWCE repartitions the data into chunks of size $s$, and retrains and re-ensembles classifiers. According to empirical evidence as detailed in Appendix B, under the condition that the chunk size should still be sufficient to avoid high classification variance, the smaller the chunk size, the lower error rate WCE gets. With the same chunk size, DWCE with a smaller buffer size can outperform WCE in accuracy. WCE and DWCE with parameter settings that have produced the lowest error rates are taken to compare with RePro.

As for CVFDT, the software published by its authors is used (http://www.cs.washington.edu/dm/vfml). There are abundant parameters. Various parameter settings are tested and the best results are taken to compare with RePro.

### 6.3 Results and analysis

The empirical results are presented and analyzed here.

**RePro, combining strength** As theoretically analyzed in Section 4, the reactive and proactive approaches each have their own pros and cons in predicting for streaming data. RePro can be a coalition of their strength. The empirical results have verified these understandings. Experiments are conducted on different Stagger data streams which possess different degrees of randomness in concept transitions[9]. With $z$ value of Zipfian distribution increasing, the concept transition changes from stochastic to deterministic. For example, the transition matrix when $z = 0.5$, $z = 1$ and $z = 7$ respectively is presented in Table 6.

| Current Concept | Next Concept | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $z = 0.5$ | | | $z = 1$ | | | $z = 7$ | | |
| | $\mathcal{A}$ | $\mathcal{B}$ | $\mathcal{C}$ | $\mathcal{A}$ | $\mathcal{B}$ | $\mathcal{C}$ | $\mathcal{A}$ | $\mathcal{B}$ | $\mathcal{C}$ |
| $\mathcal{A}$ | - | 4 | 4 | - | 5 | 1 | - | 8 | 0 |
| $\mathcal{B}$ | 2 | - | 5 | 3 | - | 7 | 0 | - | 8 |
| $\mathcal{C}$ | 5 | 4 | - | 3 | 5 | - | 8 | 0 | - |

**Table 6.** Transition matrix with different Zipfian distributions in Stagger

[9] One can not manipulate these degrees in the hyperplane or network intrusion data, for which no results are presented.

The corresponding prediction error rate and time of each modeling method are depicted in Figure 3. Contemporary-reactive (Reactive (C.)) modeling does not consult history and its performance is inordinate. It is fast but its error rate is always high. Historical-reactive (Reactive (H.)) modeling exhaustively inspects each distinct historical concept. Its prediction error is hence to some degree independent of $z$ value and is always low. But its prediction time is in general longer than proactive modeling. Proactive modeling is more sensitive to the determinism of concept transitions. Hence its error rate and time decrease as $z$ value increases. RePro can achieve a rapport between proactive and reactive modelings' efficiency and efficacy. When $z$ value is very small, the transition among concepts tends to be stochastic. The proactive method almost always ends up with wrong anticipations and hence incurs a high error rate. RePro resorts to its reactive component when finding no strong candidate. This consultation makes RePro slower than purely proactive. But it lowers the error rate. When $z$ value grows bigger and patterns of concept transition come into being, RePro takes advantage of its proactive component and achieves better efficiency compared with reactive modeling with little loss of prediction accuracy. Please note that the time reduction of the proactive approach compared with the reactive approach is not huge in Stagger because Stagger only has three concepts. More striking efficiency boost can be expected in data that involve more alternative concepts.

Throughout this section, unless otherwise mentioned, the Stagger data set means Stagger with $z = 1$ of the Zipfian distribution, simulating the most common case in the real world where the transitions among concepts are probabilistic (neither deterministic nor totally random).

**Conceptual equivalence, advisable strategy** As theoretically analyzed in Section 3, it is advisable to use conceptual equivalence (CE) when building a concept history. It can shorten the candidate list in reactive modeling. It also helps reveal associations among concepts and helps learn transition matrix in proactive modeling. The empirical evidence has supported these understandings.

RePro is tested in all three data sets, both with and without conceptual equivalence. In Figure 4, a brick bar represents the ratio equal to *error rate with CE* divided by *error rate without CE*. A solid bar represents the ratio equal to *prediction time without CE* divided by *prediction time with CE*. In every data set, the error increase (if any) caused by using CE is marginal (ratio $\approx$ 1), whereas the efficiency is largely boosted by using CE. The boost tends to become more significant with the history growing. These results are particularly attractive because the proposed equivalence measure circumvents syntactically comparing instances or classification rules of streaming data. Take Hyperplane as example. Two sequences of 500 instances[10] are produced by the same underlying concept. Their class distributions are also the same: 50% positive and 50% negative instances. However, since the $x_i$s are randomly produced along

[10] The sample size is chosen to avoid observation noise caused by high classification variance.

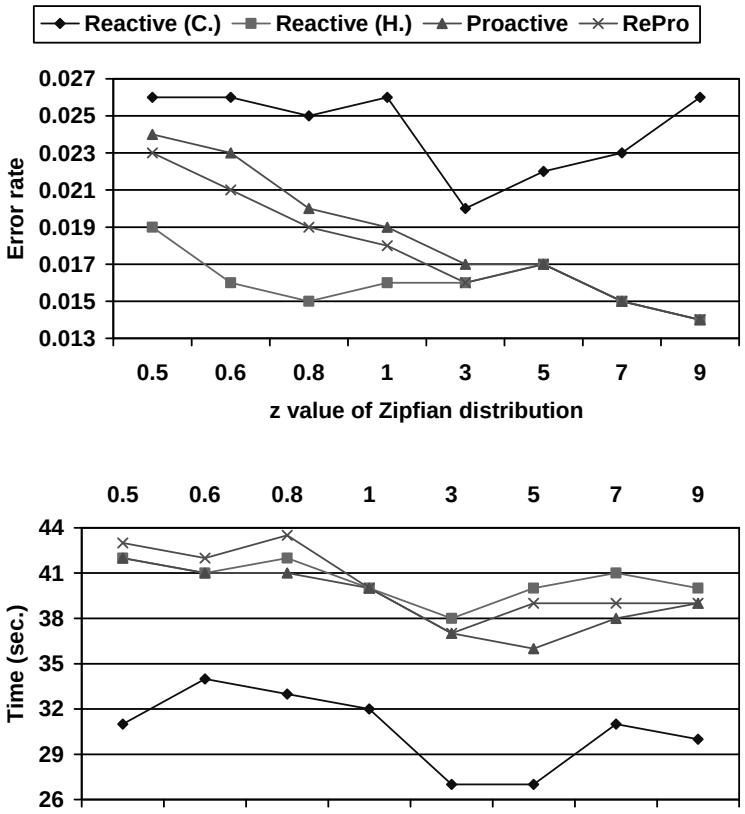



**Fig. 3.** RePro incorporates the efficacy of Reactive and the efficiency of Proactive.

time, instances between the two sequences seldom overlap. The two rule sets, each learned by C4.5rules from a sequence, have very different appearances as shown in Table 7. Without measuring conceptual equivalence, the reappearance of a concept can often be treated as a new concept, which is not advisable. In contrast, the proposed conceptual equivalence measure with a proper threshold (0.9 in this case) will successfully identify that the two concepts are equivalent.

**Comparing with rivals** As theoretically analyzed in Section 5, compared with existing representative approaches, RePro provides a more effective and efficient solution to predicting in data streams. The empirical results have supported these expectations.

*Verifying on artificial data* The prediction error rate of each method, RePro, WCE, DWCE and CVFDT, averaged on each data stream, Stagger, Hyperplane and Network Intrusion is illustrated in Figure 5, together with each method's

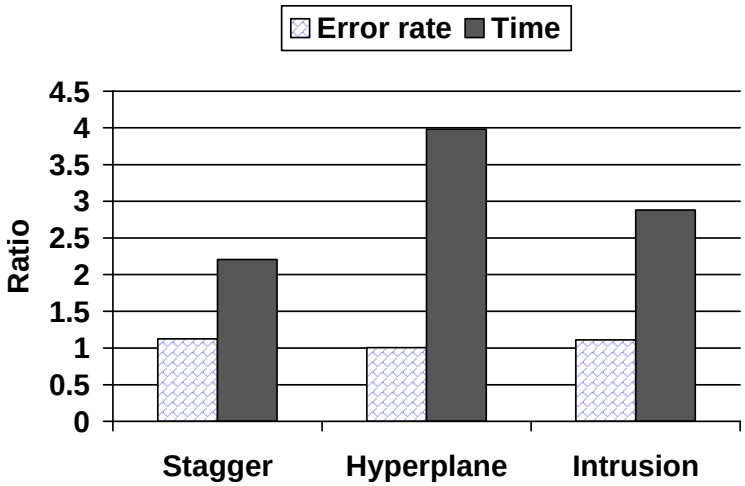



**Fig. 4.** Conceptual equivalence boosts prediction efficiency with little loss of accuracy.

total classification time. Their incremental classification error rates are depicted in Figures 6, 7 and 8, detailing their prediction performance along time.

Generally, RePro achieves lower error rates than all other methods in all data sets except for DWCE in Hyperplane, in which case RePro is far more efficient than DWCE. As a matter of fact, DWCE's time overhead is prohibitively high, making it almost unfeasible for predicting in streaming data.

Specifically in Stagger (Figure 6), the concept shifts among $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ every 500 instances as indicated by the grid lines. At the beginning, RePro's prediction error increases upon each concept change and decreases later when a proper prediction model is chosen. With time going by and the concept history growing longer, RePro learns better the transitions among concepts, adjusts faster to the concept change and achieves the lowest error rate. The second best method is CVFDT. Compared with RePro, CVFDT always learns a concept from scratch upon trigger detection no matter whether this concept is a reappearance of an old one. This rebuilding needs time and incurs high classification variance before the classifier becomes stable. Hence it has a slower adaption to the concept change and incurs more prediction errors than RePro. Although DWCE is as expected better than WCE in terms of prediction accuracy, both are trigger-insensitive that produces highest error rates in the context of concept shift.

In Hyperplane (Figure 7), the hyperplane (concept) drifts up and down. This is the niche for WCE and DWCE because the most recent concept is always the most similar to the new one. Nonetheless, RePro offers a surprisingly good prediction accuracy that is competitive with WCE and DWCE. An insight into RePro reveals that the conceptual equivalence measure distinguishes all concepts into two equivalence groups: one with $w_i \in [-3, 0]$ and the other with $w_i \in (0, +3]$. Otherwise the hyperplanes are too close to be worth differentiating. As a result, a concept sequence of $\mathcal{A}$, $\mathcal{B}$, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{A}$, $\mathcal{B}, \cdots$ is learned. To classify

**Table 7.** Syntactically comparing learned rules does not always offer a good indication to the concept reappearance.

| **Rules learned from the first sequence** |
|---|
| 1: Att1<=0.41, Att3<=0.39⇒Class=+ (100%) |
| 3: Att1<=0.28, Att3<=0.53⇒Class=+ (100%) |
| 5: Att1<=0.28, Att2>0.35, Att3<=0.66⇒Class=+ (100%) |
| ... |
| 24: Att1>0.53, Att2<=0.93, Att3>0.51⇒Class=- (97.7%) |
| 10: Att1>0.27, Att3>0.8⇒Class=- (95%) |
| default:Class=- |
| **Rules learned from the second sequence** |
| 1: Att1<=0.33, Att3<=0.47⇒Class=+ (100%) |
| 2: Att1<=0.67, Att3<=0.09⇒Class=+ (100%) |
| 4: Att1<=0.67, Att2>0.1, Att3<=0.21⇒Class=+ (100%) |
| ... |
| 26: Att1>0.37, Att3>0.78⇒Class=- (97.5%) |
| 12: Att2<=0.27, Att3>0.47⇒Class=- (97.5%) |
| default:Class=- |

an instance of a latter $\mathcal{B}$, RePro employs a former $\mathcal{B}$. That is no less effective than classifying an instance by its recent instances when the concept drifts. As for CVFDT, it incurs the highest error rate. The reason is that the concept of $\sum_{i=1}^{d} w_i x_i \geq w_0$ is not an easy one to learn and CVFDT does not consult the history. CVFDT needs to wait for a long period of time and see many labeled instances before it builds a new stable classifier from scratch, during which period many prediction mistakes have been made. This is compounded by the dilemma that the concept may soon change again after this (relatively long) waiting period and the newly-stabled classifier has to be discarded right away[11].

In Network Intrusion (Figure 8), sampling change takes place. In this particular case, transitions among different intrusion types are more random than deterministic. Hence, the reactive component of RePro often takes over the prediction task. Because an intrusion type sometimes re-appear itself in the data stream, RePro is able to reuse historical concepts and achieve the lowest prediction error rate. CVFDT acquires a competitive performance because the concept of each intrusion type is much easier to learn even if starting from scratch. Again WCE and DWCE witness suboptimal results because there does not necessarily exist continuality among intrusion types. Hence the recent concepts are not always the most appropriate to use.

[11] These error rates may sometimes be higher than those reported in the original work [3]. It is because the original work used a much larger data size. There are many more instances coming after the new classifier becomes stable and hence can be classified correctly. This longer existence of each concept relieves CVFDT's dilemma and lowers its average error rate.

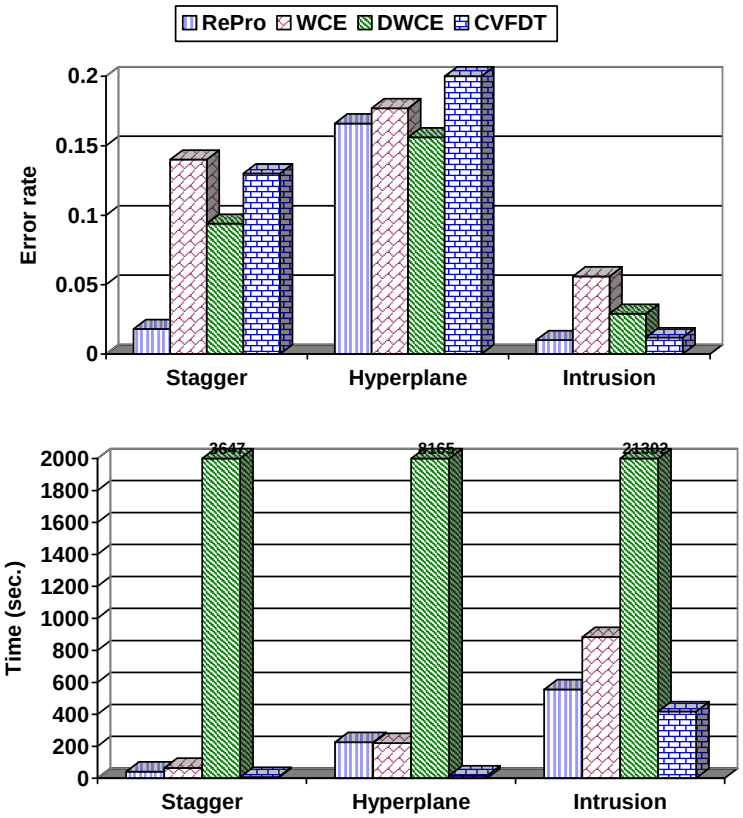



**Fig. 5.** Performance comparison in prediction efficacy and efficiency.

In summary, RePro excels in scenarios of concept shift and sampling change. It also offers a competitive accuracy when concept drifts. In all cases, RePro is very efficient.

*Verifying on real-world data* The prediction error rate of each method, RePro, WCE, DWCE and CVFDT, averaged on each data stream, is listed in Table 9. Each method's total classification time is listed in Table 10. Figure 9 illustrates several representative incremental learning curves of rival algorithms.

To statistically compare algorithms, a win/lose/tie record is calculated for each pair of them, say $A$ and $B$, across the 36 data streams. The record represents the number of data streams in which $A$ beats or loses to or ties with $B$ respectively, with regard to their prediction error rates. A one-tailed binomial sign test can be applied to the record. If its result is no bigger than 0.05, one may deem that the wins against losses are statistically significant and hence $A$ has a systematic (instead of by chance) advantage over $B$. The records are listed in Table 8.

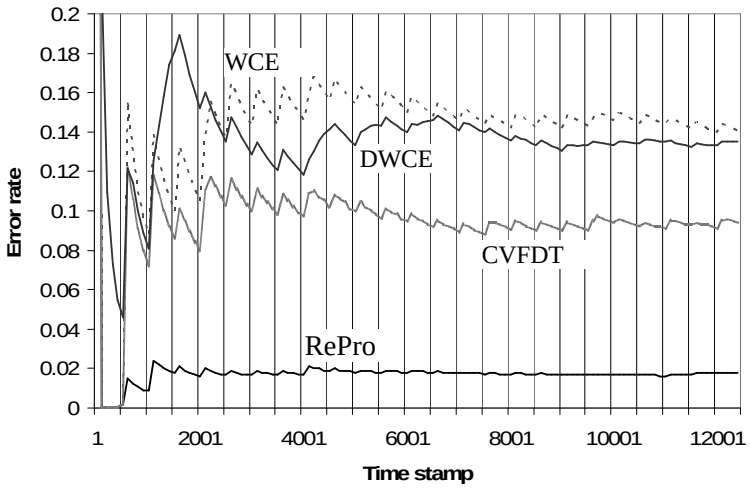



**Fig. 6.** Incremental prediction error rate on Stagger (concept shift)

In terms of prediction accuracy, RePro is statistically significantly better than other methods at the critical level 0.05. CVFDT is the second best. Both WCE and DWCE are sub-optimal with DWCE being better than WCE. In terms of prediction time, CVFDT is the fastest. RePro is the second best. WCE is slower and DWCE is the slowest. Those observations in general consist with the observations on artificial data streams.[12]

| Method | RePro | WCE | DWCE | CVFDT |
|---|---|---|---|---|
| RePro | 0/0/36 | **26/6/4** | **24/7/5** | **23/6/7** |
| WCE | 6/26/4 | 0/0/36 | 3/23/10 | 13/23/0 |
| DWCE | 7/24/5 | **23/3/10** | 0/0/36 | 13/23/0 |
| CVFDT | 6/23/7 | 23/13/0 | 23/13/0 | 0/0/36 |

**Table 8.** Win/lose/draw record of classification error rate across real-world data streams when each method in column comparing with each in row. A **bold face** record indicates that the wins against losses are statistically significant using a one-tailed binomial sign test at the critical level 0.05.

[12] Please note that for DWCE, the optimal version whose buffer size equals to 10% of its window size has been used on those 3 artificial data streams. However, its prohibitively high time demand makes DWCE intractable when a large number (36) of real-world data streams are tested here. Hence a compromise version is used here instead whose buffer size is half of its window size. The results are sufficient to verify that DWCE trades time for accuracy. It can improve prediction accuracy on WCE, but is often too slow to be useful for on-line prediction.

| Data Stream | RePro | WCE | DWCE | CVFDT |
|---|---|---|---|---|
| anneal.nominal | 21.8 | 21.9 | 21.8 | 23.3 |
| anneal.numeric | 29.0 | 24.5 | 25.1 | 24.0 |
| anneal.class | 8.3 | 36.5 | 31.7 | 8.0 |
| breast-cancer-wisconsin.nominal | 16.1 | 30.7 | 25.6 | 18.5 |
| breast-cancer-wisconsin.nominal | 29.7 | 33.4 | 33.2 | 30.6 |
| breast-cancer-wisconsin.class | 2.8 | 33.5 | 21.9 | 2.8 |
| chess.nominal | 28.7 | 25.6 | 25.6 | 35.3 |
| chess.nominal | 26.9 | 26.3 | 25.8 | 40.8 |
| chess.class | 3.7 | 38.2 | 29.2 | 3.6 |
| crx.nominal | 26.0 | 31.0 | 29.0 | 47.2 |
| crx.numeric | 35.4 | 35.9 | 30.3 | 48.7 |
| crx.class | 2.9 | 28.2 | 22.7 | 2.9 |
| heart.nominal | 37.7 | 56.2 | 52.5 | 46.1 |
| heart.numeric | 28.3 | 58.6 | 54.6 | 36.2 |
| heart.class | 7.4 | 67.6 | 55.6 | 11.9 |
| hepatitis.nominal | 20.8 | 20.8 | 20.8 | 30.3 |
| hepatitis.numeric | 20.9 | 21.9 | 21.5 | 20.2 |
| hepatitis.class | 19.2 | 20.9 | 20.9 | 19.7 |
| hypothyroid.nominal | 5.5 | 5.5 | 5.5 | 5.6 |
| hypothyroid.numeric | 7.9 | 7.7 | 7.7 | 7.9 |
| hypothyroid.class | 3.6 | 32.1 | 25.1 | 3.7 |
| ionosphere.numeric | 25.7 | 48.3 | 44.2 | 33.6 |
| ionosphere.numeric | 30.1 | 48.9 | 46.5 | 36.5 |
| ionosphere.class | 11.3 | 55.1 | 42.4 | 15.3 |
| mush.nominal | 14.2 | 22.5 | 19.9 | 18.8 |
| mush.nominal | 27.1 | 32.6 | 28.7 | 46.9 |
| mush.class | 2.0 | 19.6 | 19.6 | 2.0 |
| musk.numeric | 20.4 | 14.7 | 15.0 | 15.8 |
| musk.numeric | 24.1 | 15.9 | 16.2 | 19.8 |
| musk.class | 2.0 | 19.6 | 19.6 | 2.0 |
| sick.nominal | 6.3 | 6.3 | 6.3 | 22.7 |
| sick.numeric | 6.2 | 6.2 | 6.2 | 27.3 |
| sick.class | 2.7 | 27.1 | 20.6 | 2.7 |
| sign.numeric | 3.6 | 17.6 | 14.1 | 8.9 |
| sign.numeric | 3.9 | 17.3 | 14.4 | 4.9 |
| sign.class | 2.0 | 19.6 | 19.6 | 2.0 |
| Mean | 15.7 | 28.6 | 25.5 | 20.2 |
| GeoMean | 10.7 | 24.4 | 22.2 | 13.3 |

**Table 9.** Prediction error rate (%) of each method averaged across each data stream. *Data-set-name.[nominal|numeric|class]* indicates that the data stream is produced corresponding to a nominal attribute or a numeric attribute or the class. For details see Section 6.1. The same practice of naming applies throughout the rest of the paper.

| Data Stream | RePro | WCE | DWCE | CVFDT |
|---|---|---|---|---|
| anneal.nominal | 49 | 44 | 223 | 8 |
| anneal.numeric | 33 | 49 | 263 | 8 |
| anneal.class | 27 | 27 | 133 | 6 |
| breast-cancer-wisconsin.nominal | 14 | 10 | 57 | 2 |
| breast-cancer-wisconsin.nominal | 16 | 10 | 57 | 2 |
| breast-cancer-wisconsin.class | 15 | 22 | 147 | 6 |
| chess.nominal | 35 | 48 | 245 | 8 |
| chess.nominal | 42 | 49 | 271 | 10 |
| chess.class | 24 | 40 | 214 | 8 |
| crx.nominal | 34 | 29 | 191 | 8 |
| crx.numeric | 37 | 35 | 197 | 8 |
| crx.class | 20 | 29 | 162 | 8 |
| heart.nominal | 26 | 23 | 131 | 8 |
| heart.numeric | 26 | 35 | 192 | 12 |
| heart.class | 20 | 20 | 113 | 6 |
| hepatitis.nominal | 1 | 19 | 70 | 2 |
| hepatitis.numeric | 4 | 41 | 203 | 8 |
| hepatitis.class | 20 | 13 | 65 | 4 |
| hypothyroid.nominal | 4 | 44 | 200 | 10 |
| hypothyroid.numeric | 16 | 41 | 199 | 8 |
| hypothyroid.class | 27 | 40 | 196 | 10 |
| ionosphere.numeric | 45 | 62 | 307 | 28 |
| ionosphere.numeric | 48 | 62 | 309 | 28 |
| ionosphere.class | 30 | 51 | 244 | 14 |
| mush.nominal | 20 | 33 | 170 | 8 |
| mush.nominal | 32 | 36 | 191 | 10 |
| mush.class | 18 | 29 | 162 | 6 |
| musk.numeric | 374 | 631 | 3078 | 116 |
| musk.numeric | 130 | 192 | 972 | 74 |
| musk.class | 238 | 621 | 2662 | 48 |
| sick.nominal | 7 | 112 | 705 | 8 |
| sick.numeric | 2 | 40 | 206 | 6 |
| sick.class | 16 | 36 | 198 | 8 |
| sign.numeric | 13 | 28 | 164 | 10 |
| sign.numeric | 15 | 28 | 164 | 12 |
| sign.class | 11 | 22 | 133 | 8 |
| Mean | 41 | 74 | 367 | 15 |
| GeoMean | 22 | 40 | 213 | 9 |

**Table 10.** Total classification time (in seconds) of each method for each data stream

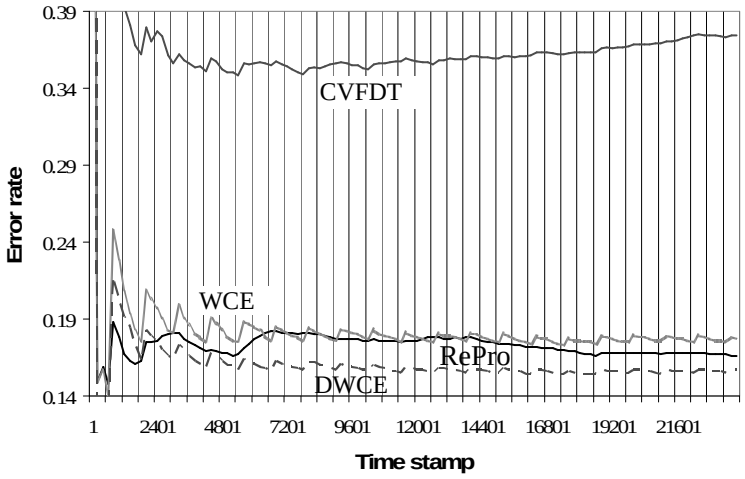



**Fig. 7.** Incremental prediction error rate on Hyperplane (concept drift)

## 7 Conclusion

Human beings live in a changing world whose history may repeat itself. Both foreseeing into the future and retrospecting into the history have been proven important in many aspects of life. This paper incorporates those useful practices in mining data streams by using a proactive-reactive prediction. In particular, this paper has proposed a novel mechanism to organize, into a concept history, data that stream along the time line. As a result, the problem of the intractable amount of streaming data is solved since concepts are much more compact than raw data while still retain the essential information. Besides, patterns among concept transitions can be learned from this history, which helps foresee the future. This paper has also proposed a novel system RePro to predict for the concept-changing streaming data. Not only can RePro conduct a reactive prediction: detecting the concept change and accordingly modifying the prediction model for oncoming instances; but also RePro can conduct a proactive prediction: foreseeing the coming concept given the current concept. By making good use of the concept history, and incorporating reactive and proactive modeling, RePro is able to achieve both effective and efficient predictions in various scenarios of concept change, including concept shift, concept drift and sampling change. Although different types of changes have different characteristics and require different optimal solutions, a challenge in reality is that one seldom knows beforehand what type of change is happening. Hence a mechanism like RePro that performs well in general is very useful.

Some further work is named below.

– The transitions among concepts can be of higher order. Currently, RePro involves first-order transitions, that is, predicting the oncoming concept given the current (single) concept. It is interesting to delve more into past concept

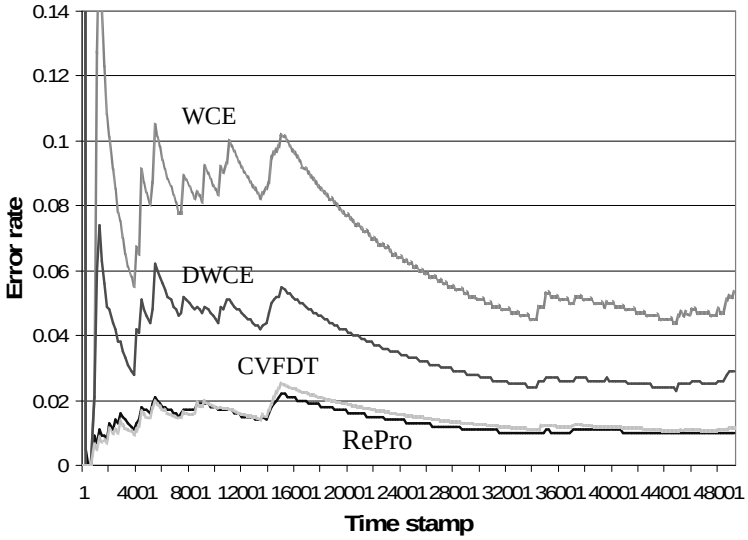


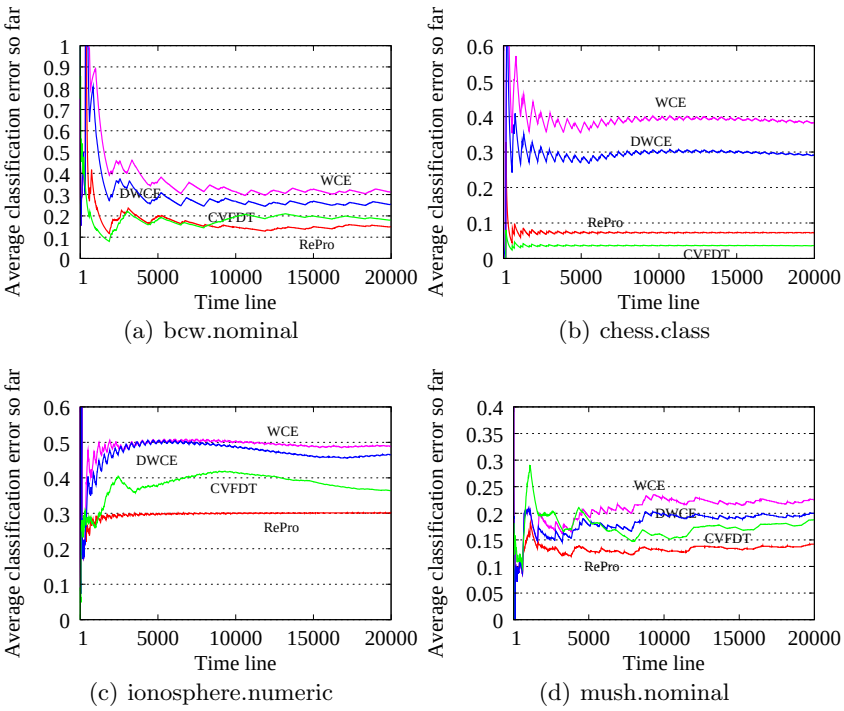

**Fig. 8.** Incremental prediction error rate on Network Intrusion (sampling change)

influences. For example, what if a sequence of concepts up to now determines the next concept? Intuitively, a longer sequence brings a more accurate prediction. However, maintaining a higher-order transition matrix is not necessarily a trivial job. The accuracy-efficiency trade-off is worth exploring.
- This paper has not focused on parameter tuning for RePro. Although it has already outperformed alternative methods without sophisticated parameter tuning, RePro may further improve by automating its parameter selection. For example, the window size, the stable learning size or the threshold of conceptual equivalence can possibly be adjusted according to different degrees of difficulty in learning a concept along the time.
- It will be useful to study how robust to noise this methodology is since noise commonly exists in streaming data.
- Different durations of a concept's existence in history may indicate different futures. For instance, a three-year drought may bring more drastic social aftermath than a half-year drought. Hence, a concept re-occurring with meaningfully different durations may be treated as distinct concepts in the history, which may have a great utility in reality. This investigation will most likely involve domain knowledge.

# APPENDIX

## A Simulating concept transition in Stagger

With the help of Zipfian distribution [16], the three concepts of Stagger can compose different sequences, simulating different scenarios among concept tran-




(a) bcw.nominal (b) chess.class

(c) ionosphere.numeric (d) mush.nominal

**Fig. 9.** Incremental prediction error rate on real-world data streams

sitions (from stochastic to deterministic). This helps get a complete picture of a prediction method's performance as if in the diversity of the real world. Two steps as in Section A.1 and A.2 compose this procedure.

Zipfian distribution is commonly observed in many phenomena [16], where the frequency of occurrence of the $n$th ranked item is $Zipf(n) = \frac{1}{n^z}$ and $z(\geq 0)$ is a parameter controlling the skewness of the distribution. The probability of occurrence of items starts high and tapers off. The bigger $z$ is, the fewer items that occur very often.

### A.1 Producing transition matrix

To produce a transition matrix of a concept, say $\mathcal{A}$, firstly the remaining concepts $\mathcal{B}$ and $\mathcal{C}$ are randomly ordered. The first one acquiring the highest rank as 1. For example, rank($\mathcal{C}$)=1 and rank($\mathcal{B}$)=2. Secondly, each concept's Zipfian value is calculated using its rank. For example, $Zipf(\mathcal{C}) = \frac{1}{1^z}$ and $Zipf(\mathcal{B}) = \frac{1}{2^z}$. Thirdly, the Zipfian value of each concept is normalized so that they sum to 1. For example $\mathcal{C}$ is associated with 0.75 and $\mathcal{B}$ with 0.25. These values 0.75 and 0.25 correspond to the probabilities of $\mathcal{A}$ transiting to $\mathcal{C}$ and $\mathcal{B}$ respectively in the transition matrix. The smaller $z$ is, the smaller the difference among the probabilities, and the more stochastic the concept transition becomes. Hence one can simulate different scenarios by changing $z$.

Once the transition matrix is built, one can accordingly produce a sequence of concepts.

### A.2 Producing concept sequence

First, randomly pick up a concept, say $\mathcal{A}$, to begin the sequence with. Then, produce a random number that is uniformly distributed in [0,1). If $rand \in [0, 0.75)$, the next concept is $\mathcal{C}$. If $rand \in [0.75, 1)$, the next concept is $\mathcal{B}$. As a result, the probability of $\mathcal{C}$ coming after $\mathcal{A}$ is 75% while $\mathcal{B}$ after $\mathcal{A}$ is 25%, which consists with the transition matrix.

Once the second concept is selected, the same selection procedure is applied to it to produce the third concept. The procedure goes on and on, building a sequence of concepts.

A number of instances can be produced for each concept in the sequence, composing a data stream.

## B WCE and DWCE

Empirical observations here supplement the discussion about WCE and DWCE in Section 6.2.

As illustrated in Figure 10 for WCE, under the condition that the chunk size should still be sufficient to avoid high classification variance, the smaller the chunk size, the lower the prediction error. One possible reason is that the smaller

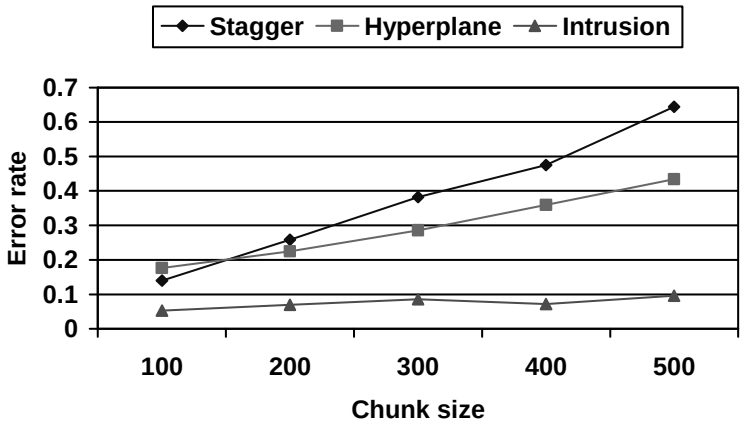



**Fig. 10.** For WCE, a smaller chunk size leads to a lower prediction error.

the chunk size, the less likely that a chunk involves instances from different concepts, and the less inordinate the ensemble classifiers may be.

As illustrated in Figure 11, compared with WCE of the same chunk size, for example size=100, DWCE achieves lower error rates. The smaller the buffer size, the more frequently DWCE adapts the ensemble to new instances, hence the lower the error rate whereas the higher the time overhead. This trade-off may not be desirable for streaming data since the loss of efficiency tends to overwhelm the gain of efficacy.

## References

1. Ganti, V., Gehrke, J., Ramakrishnan, R.: Demon: Mining and monitoring evolving data. IEEE Transactions on Knowledge and Data Engineering **13** (2001) 50–63
2. Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W.Y.: Boat-optimistic decision tree construction. In: Proceedings ACM SIGMOD international conference on Management of data. (1999) 169–180
3. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2001) 97–106
4. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2002) 102–111
5. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: A new ensemble method for tracking concept drift. In: Proceedings of the 3rd International IEEE Conference on Data Mining. (2003) 123–130
6. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2001) 377–382

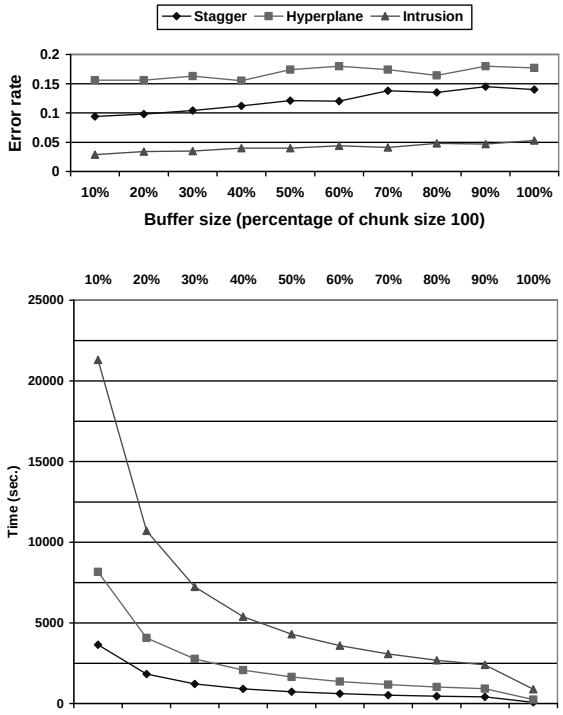



**Fig. 11.** For DWCE, a smaller buffer size decreases prediction error but shoots up prediction time.


7. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (2003) 226–235
8. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning **23** (1996) 69–101
9. Stanley, K.O.: Learning concept drift with a committee of decision trees (2003) Technical Report AI-03-302, Department of Computer Sciences, University of Texas at Austin.
10. Tsymbal, A.: The problem of concept drift: definitions and related work (2004) Technical Report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin.
11. Lanquillon, C., Renz, I.: Adaptive information filtering: Detecting changes in text streams. In: Proceedings of the 8th International Conference on Information and Knowledge Management. (1999) 538–544
12. Salganicoff, M.: Tolerating concept and sampling shift in lazy learning using prediction error context switching. Artificial Intelligence Review **11** (1997) 133–155
13. Harries, M.B., Horn, K.: Learning stable concepts in a changing world. In: PRICAI Workshops. (1996) 106–122
14. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)

15. Yang, Y., Wu, X., Zhu, X.: Dealing with predictive-but-unpredictable attributes in noisy data sources. In: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases. (2004) 471–483
16. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley-Interscience, NY (April 1991) Winner of '1991 Best Advanced How-To Book, Systems' award from the Computer Press Association.
17. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases. (2003) 81–92
18. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/mlrepository.html] (2005) Department of Information and Computer Science, University of California, Irvine.