Proactive-Reactive Prediction for Data Streams

Ying Yang, Xindong Wu and Xingquan Zhu
Department of Computer Science, University of Vermont
Burlington, Vermont, USA 05405
yyang, xwu, xqzhu@cs.uvm.edu

ABSTRACT

Prediction in streaming data is an important activity in various branches of science such as sociology, economics and politics. Two major challenges offered by data streams are (1) the underlying concept of the data may change over time; and (2) the data may grow without limit so that it is difficult to retain a long history of raw data. Previous research has mainly focused on manipulating relatively recent data. The distinctive contribution of this paper is in three folds. First, it uses a measure of *conceptual equivalence* to organize the data history into a history of concepts. Transition patterns among concepts can be learned from this history to help prediction. Second, 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. Third, it proposes a system *RePro* that incorporates *re*active and *pro*active mechanisms to predict in streaming data with efficacy and efficiency. 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 suggests the proposed methodology is an advisable solution to prediction for data streams.

1. INTRODUCTION

Prediction for data streams is important. Streaming data are very common in today's world such as Internet event logs, stock market transitional flows, or sensor network reports. Streaming data have two special characteristics. First, they are time-series data whose underlying concept may change over time. Second, the data may grow without limit and it is difficult to retain their complete history. 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.

Prediction for data streams is, however, not a trivial task. The special characteristics of streaming data have brought new challenges to the prediction task. The underlying concept change requires 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. This complication is compounded by the fact that the prediction has to be made with limited access to the history of streaming data.

Various great ideas have been contributed to prediction for data streams [2, 3, 5, 7, 8, 13, 15, 16]. With all due respect to previous achievements, it is suggested that some problems 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 be retained. As to 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 to 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*. It will help prepare for the future and can instantly launch a new prediction strategy upon detecting a concept change.

This paper sets out to tackle those open problems. The goals involve (1) organizing the history of raw data into a history of concepts. A concept represents compact and essential knowledge abstracted from raw data. One can easily keep a long history of concepts; (2) learning from the concept history patterns of concept transition, which may offer valuable clues to proactive as well as reactive prediction; and (3) a system that effectively and efficiently predict for streaming data. In particular, 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 build a history of changing concepts. Key components involve a trigger detection method, an equivalence measure named *conceptual equivalence* and a transition matrix accumulated over time. This concept history will participate in prediction. Section 4 introduces a system RePro that reactively as well as proactively predicts for streaming data, utilizing the concept history. 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

Several terms in this paper are inherited from the terminology 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. In the context of this paper, a concept is a set of rules learned by a classification algorithm. However, the proposed mechanisms are applicable regardless of the learning algorithm variations, and are hence portable.

2.2 Modes of concept change

A theme issue in streaming data research is the concept change problem. Different modes of change have been mentioned in the literature[1] as follows:

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

Concept change refers to the change of the underlying concept over time. **Concept drift** describes the *gradual* change of the concept [12, 14, 16]. For example, a slow wearing piece of factory equipment might cause a gradual change in the quality of output parts. **Concept shift** [9, 12] happens when the change between two concepts is more categorical. For example, in the history of US politics, the policy changes whenever a different party takes the government. There are not necessarily gradual transitions between a republican government and a democrat government. **Sampling change**, also known as sampling shift [11] or virtual concept drift [16], refers to the change of 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 may no longer be acceptable with the new data distribution.

[1]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 and vice versa. The definitions here are cleared up to the best of the authors' understanding.

Stanley [12] 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 5.4 is more close to 5.5 than 5.3 is. It is not meaningful in the same way to say that a piano is more similar to a violin than a saxophone is. Analogously, one may assume that the current concept is always the most similar to the oncoming concept in concept drifting because of its gradual changing [15], 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 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, for prediction. 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 [7]. Trigger-insensitive approaches do not explicitly detect triggers. Instead, they continuously adapt the current model to newly coming instances [5].

Incremental *vs.* **Batch**. Incremental approaches process coming instances one by one. Batch approaches exam a batch of instances at once [4, 14, 15].

Historical *vs.* **Contemporary**. After a trigger is detected, historical approaches resort to the history to construct a new model while contemporary approaches only consult 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 history of changing concepts in streaming data. 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 only a bunch of abstract rules and 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 the system of 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 [10] 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 a smooth transition and hence the immediate previous concept may not be the best simulator of the current 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. [7, 14] have reviewed abounding trigger detection algorithms. A *sliding-window* methodology is used here. Two important parameters are the *window size* and the *error threshold*. One by one, the current prediction model predicts the class of coming instances. The beginning of the window is always a wrongly classified 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 wrongly classified instance (if there is any) and the previous starting instance is dropped from the window.

3.3 A measure of conceptual equivalence

This measure checks whether a concept repeats itself at different times. Given current data and the corresponding learned concept, this measure calculates the degree of equivalence between this concept and each distinct historical concept, as in Table 1. If the measure between a newly learned concept V and a previous concept T is above a user-defined threshold, the system deems that V is a reappearing T .

Table 1: Measuring conceptual equivalence

```
Input: concept $D$ newly learned from current data $T$,
previous concept $V$
Output: a numeric value $\in$ [-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
```

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*'. For example, if V classifies I into class X but V classifies I into class Y, the score will be -1 since conflict_X does not equal to conflict_Y. 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 learned rule sets often differ in their faces. 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 does not reply solely on comparing classification accuracies, which is not always indicative enough. For example, if V and T classify D with poor accuracies 75% and 65% respectively, their equivalence can be very high if they agree on classifying many instances even when the classification is wrong.

3.4 A stable learning size

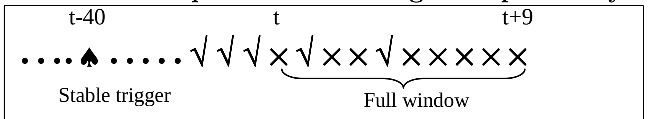
This is a parameter that specifies the number of instances above which the learned concept can be deemed stable. In order to be sensitive to concept change, the window size of trigger detection is normally small. Sometimes this window of instances are *sufficient* to indicate that the current concept 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[2], and 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_ts 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_ts are deemed as a series of stable concepts and $\overline{H}$ is updated by H. Prediction always resorts to the stable history first. 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.

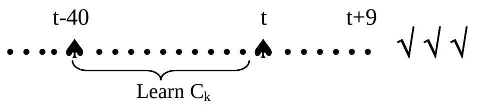
3.5 The building process

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 wrongly classified instance. Only instances coming after the last stable trigger and up to now are retained for the purpose of calculating conceptual equivalence. Other historical instances are not essential to keep.

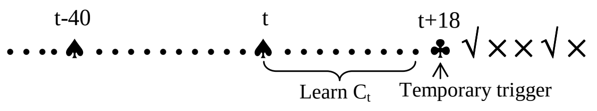
Table 2: The process of building concept history



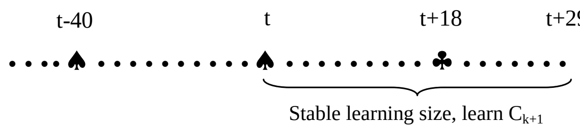
Stage 1 Starting from t, wrong prediction is witnessed. Up to t+9, the window is full and the error rate is 80% which is above the error threshold. Hence, t is detected as a new trigger.



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. Update both stable and contemporary concept histories.



Stage 3 A new prediction model is chosen to predict for instances coming after t+9. Assume a new trigger is detected at t+18. A concept C_t is learned over [t, t+18), which 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, 30 instances are known. If no (temporary) trigger happens in [t, t+29], do nothing. Otherwise, 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 stable history and update temporary history by stable history. Otherwise update stable history by temporary history.

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 not proper any more. A different model needs to be chosen to classify oncoming instances. Available information to make this choice is: (1) a history of concepts up to this triggers; 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 Reactive

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

4.1.1 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 seldom 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 emergency if the concept is brand new and has no history to learn from.

4.1.2 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

[2]For example, it is of high classification variance by overfitting a small portion of instances.

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 also 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.

4.2 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 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 [6]. 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 occurrence of a 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, Frances, Ivan and Jeanne during Year 2004 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.* A transition matrix can be built and updated along the time. Table 3 shows the final status of a first-order transition matrix[3].

Table 3: Transition Matrix

	Next State					
Current 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

Suppose that the current concept is autumn. According to the transition matrix, in history the most frequent concept after autumn is winter. Hence, 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 is chosen in advance instead of upon trigger detection; and the reaction time to concept change is short since there is no need to validate historical concepts one by one as in the reactive approach. A problem happens when there is not a 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 so forth. 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.

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

4.3 RePro, coalition of strength

As addressed above, reactive and proactive modeling each excels in different scenarios. A system RePro (reactive plus proactive), as in Table 4, incorporates them together and uses one's strength to offset another's weakness.

As a result, if the proactive component of RePro foresees multiple probable concepts given the current concept, one can use the 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 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 in Section 3.4, 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. Two typical mechanisms are discussed here. More comprehensive reviews can be found in various informative survey papers [7, 14].

The FLORA system [16] 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 [5]. It represents concepts by conjunctions of attribute values and measures the conceptual equivalence by syntactical comparison. This is less applicable in streaming data, as will be demonstrated in Section 6.3.2. The concepts are stored only for the reactive purpose. FLORA does not explore their associations and hence can not be proactive.

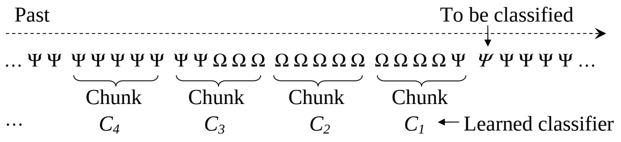
The weighted classifier ensemble (WCE) approach [15] rep-

Table 4: RePro: reactive+proactive

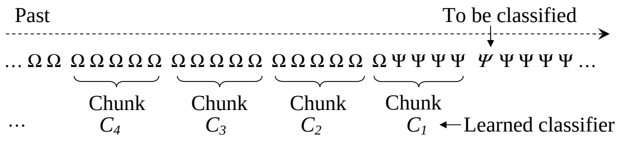
```
Input: A concept history containing a list of distinct historical concepts $C\_DIS$, a concept sequence $C\_SEQ$ and a transition matrix $C\_TRA$. A window of instances $I\_WIN$ that has just triggered the concept change. A probability threshold $threshold_{prob}$. A classification accuracy threshold $threshold_{accu}$
Output: a prediction model to predict for oncoming instances.
Begin
$c_{last}$ = the last stable concept in $C\_SEQ$;
// Proactive
$c_{probable}$(s) = concept(s) whose probability given $c_{last}$ is bigger than $threshold_{prob}$, according to $C\_TRA$;
IF a single $c_{probable}$ exists
  return $c_{probable}$;
IF multiple $c_{probable}$'s exist
  // Reactive historical
  FOREACH $c_{probable}$
    calculate its accuracy on $I\_WIN$;
  IF the highest accuracy is bigger than $threshold_{accu}$
    return $c_{probable}$ acquiring highest accuracy;
  ELSE
  // Reactive contemporary
    return the concept learned from $I\_WIN$;
IF no $c_{probable}$ exists
  FOREACH concept $c_{historical} \in C\_DIS$
    calculate its accuracy on $I\_WIN$;
  IF the highest accuracy is bigger than $threshold_{accu}$
    return $c_{historical}$ acquiring highest accuracy;
  ELSE
  // Reactive contemporary
    return the concept learned from $I\_WIN$;
End
```

resents a big family of algorithms that ensemble learners for prediction. In this work, it is proved that a carefully weighted 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 weighted 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 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 Ψ is to be classified. Because the majority of instances in chunk C_1 are of concept Ω, classifiers suitable for Ω (such as C_1 and C_2) will be ironically given a higher priority than those suitable for Ψ (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 had a long run and dominates the recent history, consulting this history does not necessarily help prediction. As illustrated below, when classifying an instance of concept Ψ, most classifiers are still indulged in the previous concept Ω. As a result, even when the most recent data chunk favors instances of Ψ, its correct weighing can be overshadowed by the large number of improper classifiers (a single high-weight C_4 against many low-weight C_1, C_2 and C_3s).



The concept-adapting very fast decision tree (CVFDT) [5] is one of the most well-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.

Table 5: Taxonomy of methods

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

In the following section of experimental evaluations, empirical comparisons will be conducted among RePro, WCE and CVFDT to verify the above understandings [4].

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 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 that they have private access to. A potential problem is that these private data sets are seldom reusable by other researchers because of organization policies. In their inspiring paper, Keogh and Kasetty have advocated using benchmark data sets that allow public access and allow fair comparisons among rival methods [7]. Kolter and Maloof agreed with this ideology by using only benchmark data sets in their experiments [8]. This paper as well employs three benchmark data sets covering all types of concept changes that have been studied in Section 2.2: concept drift, concept shift and sampling change. When applicable, the instances are randomly generated taking time as the seed. Hence, the reappearance of concepts does not necessarily mean the reappearance of instances, which better simulates the real world.

6.1.1 Hyperplane

This data set can simulate the scenario of *concept drift* by continuously moving a hyperplane [5, 8, 13, 14, 15]. 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.

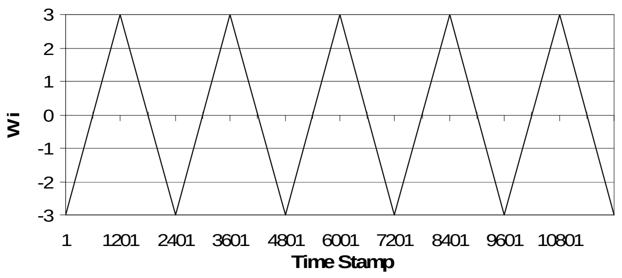


Figure 1: To simulate concept drift, a hyperplane gradually moves by continuously changing each w_i. Particularly in this experiment, the changing range is [-3,+3] and the changing rate is 0.005 per instance.

6.1.2 Stagger

This data set can simulate the scenario of *concept shift* [8, 12, 14, 16]. 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.

6.1.3 Network intrusion

This is another public accessible streaming data set and can simulate the scenario of *sampling change* [1]. 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 (Class 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

[4]FLORA is not involved since it is not oriented to handling streaming data's large amount.

well classify each type of intrusion[5]. 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.

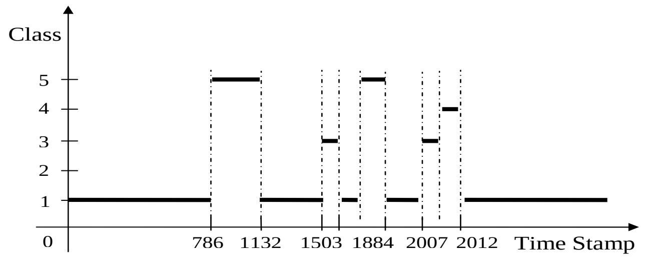


Figure 2: Network intrusion is typical sampling change. Class 1, 2... are not numeric but nominal.

6.2 Rival methods

As discussed in Section 5, two representative methods for prediction in streaming data, the weighted classifier ensemble (WCE) [15] and the concept-adapting very fast decision tree (CVFDT) [5], 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 $f < s$). Instead of waiting for a full chunk, once f new instances are labeled, DWCE repartitions data into chunks of size s, and retrain and re-ensemble classifiers. According to empirical evidence as detailed in Appendix B, the smaller the chunk size, the lower error 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.

6.3.1 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[6]. With the 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.

Table 6: Transition matrix with different Zipfian distributions in Stagger

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

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 the 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 transitions among concepts. Hence its error rates and time decrease as the z value increases. RePro makes use of reactive modeling to reduce mistakes while takes advantage of proactive modeling to speed up prediction. Hence, it can achieve a rapport between efficiency and efficacy. Please be noted 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 the following paper, 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, and neither deterministic nor totally random.

6.3.2 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. Presented 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

[5]For example, C4.5rules [10] can achieve a 100% classification accuracy on the whole data set.

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

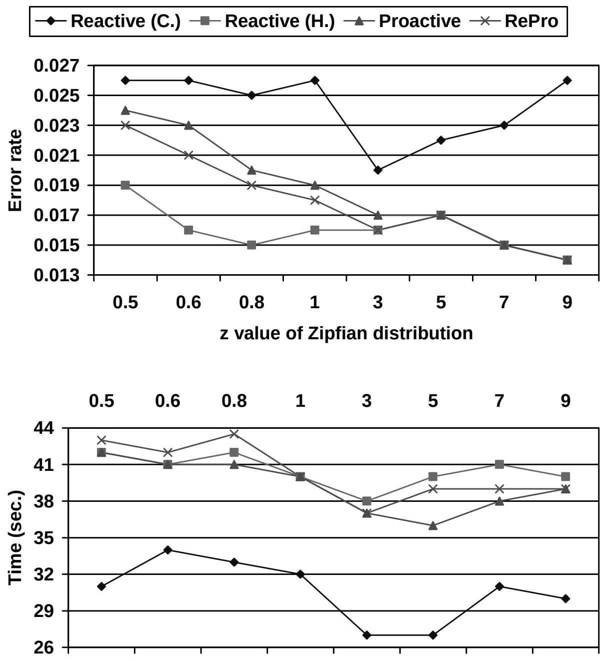


Figure 3: RePro incorporates the efficacy of Reactive and the efficiency of Proactive.

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 ≈ 1), whereas the efficiency is largely boosted by using CE. The boost tends to become more significant with the history growing.

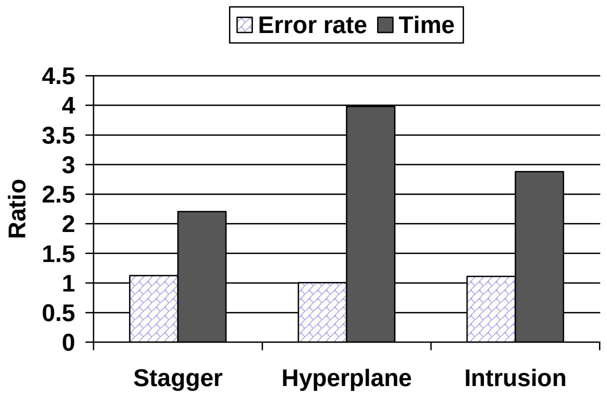


Figure 4: Conceptual equivalence boosts prediction efficiency with little loss of accuracy.

These results are particularly attractive because the proposed equivalence measure circumvents syntactically comparing instances or concepts in streaming data. Take Hyperplane as example. Two sequences of 500 instances[7] are produced by the same underlying concept. Their class distributions are also the same: 50% positive and 50% negative instances. However, since the x_is are randomly produced along time, instances between the two sequences seldom overlap. Neither do the rule sets learned by C4.5rules from each sequence repeat each other as in Table 7. Accordingly, without conceptual equivalence, the reappearance of a concept can often be treated as two different concepts, 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.

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

6.3.3 *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.

The prediction error and time of each method, RePro, WCE, DWCE and CVFDT, on each data set, Stagger, Hyperplane and Network Intrusion are illustrated in Figure 5. The incremental reports are depicted in Figure 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.

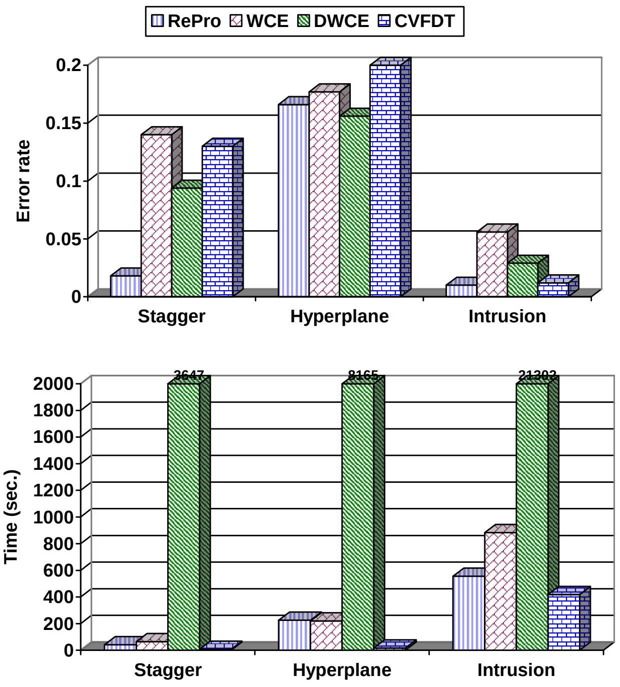


Figure 5: Performance comparison in prediction efficacy and efficiency.

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

Rules learned from the first sequence		**Rules learned from a second sequence**	
1:	Att1<=0.41, Att3<=0.39⇒Class=+ (100%)	1:	Att1<=0.33, Att3<=0.47⇒Class=+ (100%)
3:	Att1<=0.28, Att3<=0.53⇒Class=+ (100%)	2:	Att1<=0.67, Att3<=0.09⇒Class=+ (100%)
5:	Att1<=0.28, Att2>0.35, Att3<=0.66⇒Class=+ (100%)	4:	Att1<=0.67, Att2>0.1, Att3<=0.21⇒Class=+ (100%)
14:	Att1<=0.8, Att2>0.32, Att3<=0.24⇒Class=+ (100%)	6:	Att1<=0.39, Att2>0.18, Att3<=0.47⇒Class=+ (100%)
17:	Att1<=0.72, Att2>0.55, Att3<=0.51⇒Class=+ (100%)	8:	Att1<=0.67, Att2>0.38, Att3<=0.47⇒Class=+ (100%)
18:	Att2>0.55, Att3<=0.23⇒Class=+ (100%)	13:	Att1<=0.28, Att2>0.27, Att3<=0.73⇒Class=+ (100%)
20:	Att1<=0.93, Att2>0.68, Att3<=0.51⇒Class=+ (100%)	17:	Att2>0.55, Att3<=0.38⇒Class=+ (100%)
23:	Att1<=0.53, Att1>0.43, Att2>0.76⇒Class=+ (100%)	19:	Att2>0.9, Att3<=0.53⇒Class=+ (100%)
25:	Att1<=0.85, Att2>0.93⇒Class=+ (100%)	22:	Att1<=0.64, Att2>0.77, Att3<=0.78⇒Class=+ (100%)
8:	Att1<=0.43, Att2>0.41, Att3<=0.8⇒Class=+ (98.7%)	23:	Att1<=0.74, Att2>0.55, Att3<=0.56⇒Class=+ (100%)
12:	Att1<=0.73, Att2>0.19, Att3<=0.24⇒Class=+ (98.3%)	20:	Att1<=0.37, Att2>0.55⇒Class=+ (98.4%)
9:	Att1<=0.27, Att2>0.41⇒Class=+ (98.3%)	10:	Att2>0.37, Att3<=0.15⇒Class=+ (97.5%)
7:	Att1>0.28, Att2<=0.41, Att3>0.39⇒Class=- (100%)	3:	Att1>0.33, Att2<=0.1, Att3>0.09⇒Class=- (100%)
11:	Att1>0.43, Att2<=0.19⇒Class=- (100%)	7:	Att1>0.39, Att2<=0.38, Att3>0.21⇒Class=- (100%)
13:	Att1>0.73, Att2<=0.32⇒Class=- (100%)	9:	Att1>0.67, Att2<=0.37⇒Class=- (100%)
15:	Att1>0.8, Att2<=0.55⇒Class=- (100%)	11:	Att1>0.67, Att2<=0.55, Att3>0.15⇒Class=- (100%)
16:	Att1>0.43, Att2<=0.55, Att3>0.24⇒Class=- (100%)	14:	Att2<=0.55, Att3>0.73⇒Class=- (100%)
19:	Att1>0.72, Att2<=0.68, Att3>0.23⇒Class=- (100%)	15:	Att1>0.28, Att2<=0.55, Att3>0.47⇒Class=- (100%)
21:	Att1>0.93, Att3>0.23⇒Class=- (100%)	18:	Att1>0.86, Att2<=0.9, Att3>0.38⇒Class=- (100%)
26:	Att1>0.85, Att3>0.51⇒Class=- (100%)	25:	Att1>0.64, Att3>0.56⇒Class=- (100%)
4:	Att2<=0.35, Att3>0.53⇒Class=- (98.1%)	21:	Att1>0.37, Att2<=0.77, Att3>0.53⇒Class=- (97.6%)
6:	Att2<=0.41, Att3>0.66⇒Class=- (97.9%)	26:	Att1>0.37, Att3>0.78⇒Class=- (97.5%)
24:	Att1>0.53, Att2<=0.93, Att3>0.51⇒Class=- (97.7%)	12:	Att2<=0.27, Att3>0.47⇒Class=- (97.5%)
10:	Att1>0.27, Att3>0.8⇒Class=- (95%)		default:Class=-
	default:Class=-		

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 can achieve 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 rebuild 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 which is inadvisable and 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 data are always the most similar to the new ones. Nonetheless, RePro offers surprisingly good prediction accuracy that is competitive with WCE and DWCE. An insight into RePro reveals that the conceptual equivalence measures all concepts with $w_i \in [-3, 0]$ as equivalent and all with $w_i \in (0, +3]$ as equivalent. 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 an instance of a latter $\mathcal{B}$, RePro employs a former $\mathcal{B}$. This 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 at that time[8].

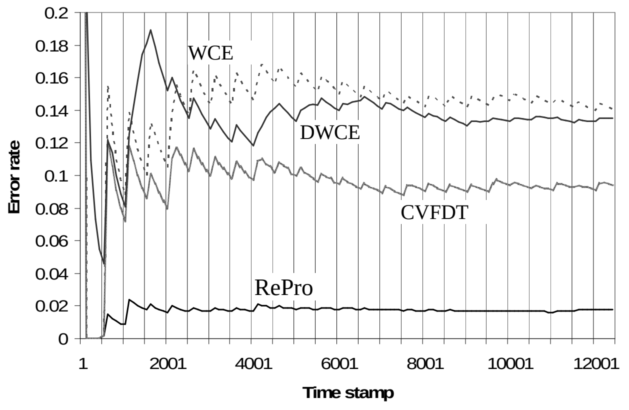


Figure 6: Incremental report on Stagger (concept shift)

[8]These error rates may sometimes be higher than those reported in the original work [5]. 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 surmounts the misclassifications and the average error rate is hence lower.

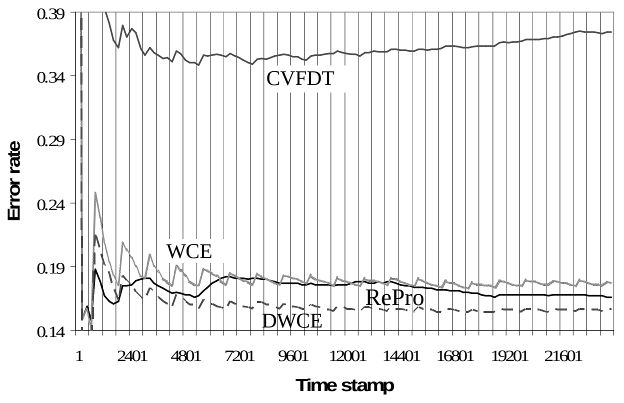


Figure 7: Incremental report on Hyperplane (concept drift)

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. CVFDT acquires 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.

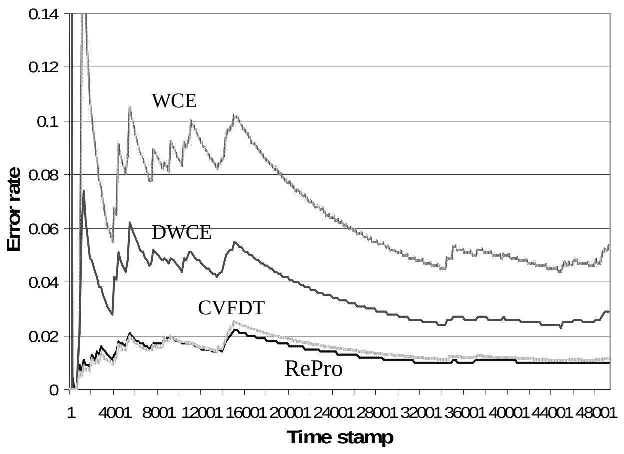


Figure 8: Incremental report on Network Intrusion (sampling change)

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

7. CONCLUSION

Human beings live in a changing world whose history may repeat itself. Hence, both foreseeing into the future and retrospecting into the history are important. This paper has proposed a novel mechanism to organize, into a concept history, data that stream in the time space. 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 grasp 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.

Both theoretical analysis and empirical evidence here have presented a framework into this new direction of reactive-proactive prediction. Some further work is named bellow.

- 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 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 difficulties of learning a concept along the time.

- 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 reoccurring 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.

In summary, its encouraging present and inspiring future suggest that the proposed proactive-reactive methodology is promising to make a better sense of this changing world and to achieve a better prediction for data streams.

8. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases*, pages 81–92, 2003.

[2] V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Mining and monitoring evolving data. *IEEE Transactions on Knowledge and Data Engineering*, 13(1):50–63, 2001.

[3] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.-Y. Loh. Boat-optimistic decision tree construction. In *Proceedings ACM SIGMOD international conference on Management of data*, pages 169–180, 1999.

[4] M. B. Harries and K. Horn. Learning stable concepts in a changing world. In *PRICAI Workshops*, pages 106–122, 1996.

[5] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106, 2001.

[6] R. Jain. *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.

[7] E. Keogh and S. Kasetty. 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*, pages 102–111, 2002.

[8] J. Z. Kolter and M. A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Proceedings of the 3rd International IEEE Conference on Data Mining*, pages 123–130, 2003.

[9] C. Lanquillon and I. Renz. Adaptive information filtering: Detecting changes in text streams. In *Proceedings of the 8th International Conference on Information and Knowledge Management*, pages 538–544, 1999.

[10] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.

[11] M. Salganicoff. Tolerating concept and sampling shift in lazy learning using prediction error context switching. *Artificial Intelligence Review*, 11(1-5):133–155, February 1997.

[12] K. O. Stanley. Learning concept drift with a committee of decision trees, 2003. Technical Report AI-03-302, Department of Computer Sciences, University of Texas at Austin.

[13] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382, 2001.

[14] A. Tsymbal. The problem of concept drift: definitions and related work, 2004. Technical Report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin.

[15] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235, 2003.

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

APPENDIX

A. SIMULATING CONCEPT TRANSITION IN STAGGER

With the help of Zipfian distribution [6], the three concepts of Stagger can compose different sequences, simulating different scenarios among concept transitions (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 compose this procedure.

Zipfian distribution is commonly observed in many phenomena [6], where the frequency of occurrence of the nth 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. Hence, 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 9 for WCE, the smaller the chunk size[9], the lower the prediction error. One possible reason is that the smaller the chunk size, the less likely that a chunk involves instances from different concepts, and the less inordinate the ensemble classifiers may be.

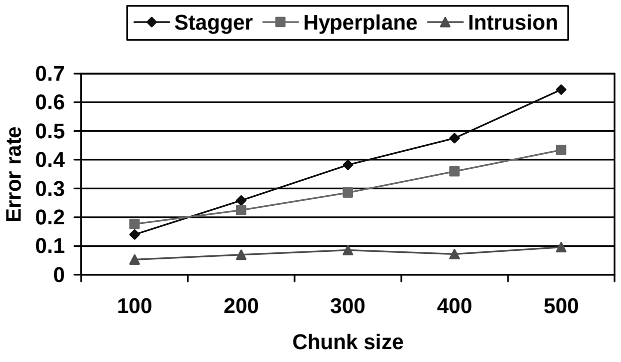


Figure 9: For WCE, a smaller chunk size leads to a lower prediction error.

As illustrated in Figure 10, 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.

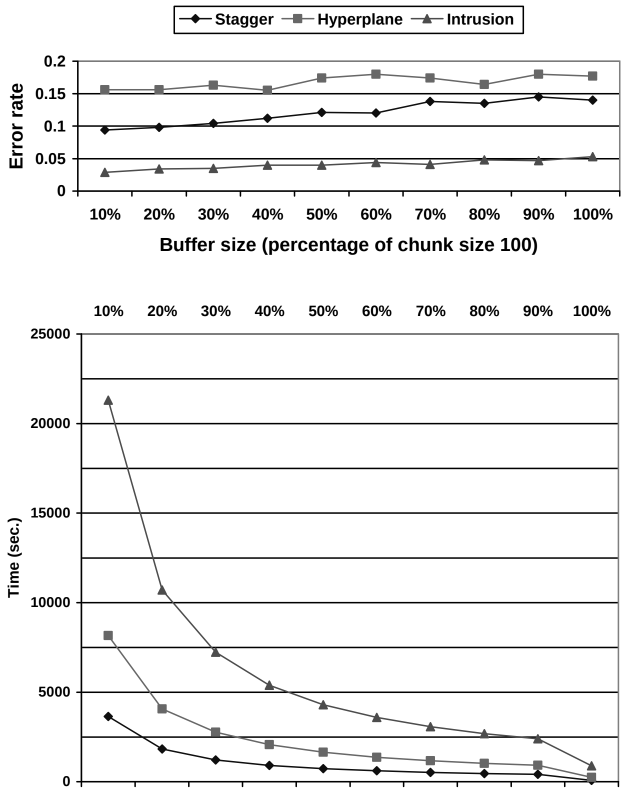


Figure 10: For DWCE, a smaller buffer size decreases prediction error but shoots up prediction time.

[9]Of course, the chunk size should still be sufficient to avoid high classification variance.