World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:10, 2009

A New History Based Method to Handle the Recurring Concept Shifts in Data Streams

Hossein Morshedlou, Ahmad Abdollahzade Barforoush

***Abstract*—**Recent developments in storage technology and networking architectures have made it possible for broad areas of applications to rely on data streams for quick response and accurate decision making. Data streams are generated from events of real world so existence of associations, which are among the occurrence of these events in real world, among concepts of data streams is logical. Extraction of these hidden associations can be useful for prediction of subsequent concepts in concept shifting data streams. In this paper we present a new method for learning association among concepts of data stream and prediction of what the next concept will be. Knowing the next concept, an informed update of data model will be possible. The results of conducted experiments show that the proposed method is proper for classification of concept shifting data streams.

***Keywords*—**Data Stream, Classification, Concept Shift, History.

I. INTRODUCTION

RECENT developments in storage technology and networking architectures have made it possible for broad areas of applications to rely on data streams for quick response and accurate decision making. Depending on data mining objectives, existing solutions in this area roughly fall into the following three categories: clustering and querying high speed data streams, association rule mining from stream data, and generating predictive models for continuous data streams [10]. Due to concept shifts in the underlying data, how to maintain a timely updated model has become one of the most challenging problems in the domain of classification. Many approaches, including both the incrementally updated classifiers [1], [2], [5] and the ensemble classifiers [3], [6] have proved that model update is a very costly process and not depends on rate of disturbance in the data.

Data streams are generated from the events of real world so existence of associations, which there are among the occurrence of events in real world, among the concepts of data streams is logical. Recurring events in real world is another important issue which causes recurring concepts in data streams. Extraction of hidden associations among the recurring concepts can be useful for prediction of subsequent concepts in data stream. If a method can properly behave when a concept shift occurs, it can adapt itself with changes of data stream and will be succeed. Such a method should be capable of learning associations, reasoning and having a proper behavior when a concept shift occurs. In the other hand, this method should behave intelligently.

In this paper we present an intelligent method for classification of data streams with concept shift. Section II introduces background knowledge. It contains basic term definitions. Section III explains the concept shift detection method and in section IV we propose a mechanism to build a history along the journey of concept changes. Section V defines intelligent behavior. Section VI shows the experiments and the results of the conducted experiments. In section VII we give concluding remarks.

II. BASIC DEFINITIONS

Data Stream: We consider data stream as a sequence of instances where each instance is a vector of attribute values. Each instance has a class label. If class label of an instance is known, it is a training instance. Otherwise it is an unlabeled instance. Set of training instances form our training data.

Concept: The term concept is more subjective than objective. Particularly in this paper, a concept is represented by a data model which is learned by a classification algorithm.

Concept Shift: Change of the underlying concept in data stream is called concept shift. After concept shift, the previously learned data model (we call it current data model) is not accurate any more and should be updated or replaced by a new data model.

Proactive Behavior: Given the current concept, Proactive behavior foresees what the forthcoming concept will be. The predicted concept will take effect once a concept shift is detected.

Reactive Behavior: Reactive behavior does not predict what concept is coming. A new data model is learned upon a concept shift is detected. How to understand the recently learned data model is similar to which one of the previously learned data models, is described in section V.

Intelligent Behavior: When a concept shift is detected, we should decide about behaving either proactively or reactively. A correct decision making about being proactive or reactive needs intelligent behavior.

Monitoring Window (MW): Monitoring Window is used to buffer the data instances. When a concept shift is detected, we should wait to receive enough instances for learning a stable data model. When Monitoring Window is full, we have

Hossein Morshedlou is with the AmirKabir University of Technology, (e-mail: morshedlou@aut.ac.ir).
Ahmad Abdollahzade Barforoush is with the AmirKabir University of Technology (e-mail: ahmad@ceit.aut.ac.ir).

sufficient instances to induce a stable data model.

Detection Window (DW): Detection Window is inside the Monitoring Window and is used to concept shift detection. Fig. 1 shows the Monitoring Window and Detection Window.

Training Data Feature (TDF): Consider sequence of instances as training data when each instance is a vector of attribute values like (a_1, …, a_m, C). Here C is class label. Assume we have k distinct classes. TDF of this training data is defined as (1).

$$\begin{array}{l} ((CF_1(1,1),CF_2(1,1),n_1,C_1),...,(CF_1(k,1),CF_2(k,1),n_k,C_k)) \\ ... \\ ((CF_1(1,m),CF_2(1,m),n_1,C_1),...,(CF_1(k,m),CF_2(k,m),n_k,C_k)) \end{array} \quad (1)$$

In (1), $CF_1(i,j)$ is the sum of squares of a_j attribute values in instances with C_i class label. $CF_2(i,j)$ is the sum of a_j attribute values in instances with C_i class label. n_i shows the number of instances with C_i class label.

III. Concept shift detection

The Sliding-window methodology is used here to detect the underlying concept shift in data stream. Two important parameters are DW size and error threshold. The current data model is used to predict the class of coming instances one by one. The beginning of the DW is always a misclassified instance. Whenever the DW is full and its error rate exceeds the error threshold, the beginning instance is taken as a trigger; otherwise, the beginning of the DW is slid to the next misclassified instance (if there is any) and the previous instances are dropped. When a concept shift occurs, the current data model is no longer appropriate for classification task and must be replaced.

IV. Building Histories

The mechanism which our method uses to build concept histories along the journey of concept change is described here. For each concept and concept shift, the method maintains a history as shown in Table I. Assume this history holds information of concept A. Using this information we know when concept A is occurred for the first time and last time, How many times A has occurred since the first time and what is the time of nth previous occurrence of A for n = 5, 10 ,…, 1280. A new occurrence of A changes this information as below:

***Last Time**: will be updated to time of the new occurrence.*
***Counter**: will be increased by one.*
***5th previous time**: 512000+ ((621000-512000)/5)*
// Assume 621000 is time of the last occurrence.
***10th previous time**: 445000+ ((621000-445000)/10)*

For example consider A and B are two known concepts and current concept is A. Our method maintains two histories for both A and B. When a concept shift from A to B occurs for the first time, our method creates a history to maintain the information of this concept shift (A to B). At next concept shifts from A to B, the method only updates the information of this history same as above.

Because of the prohibitive volume, it is very expensive to keep all the streaming data. In comparison, a data model is compact such as a bunch of abstract rules. Keeping data models of concepts is much more tractable, so our method stores data models. They can be reused in the future instead of learning from scratch. Having a concept shift, this helps to reduce the prediction time. Furthermore the associations among different concepts can be extracted using these histories. In addition of a data model, the method stores a TDF value too. Using training data, used to learn the data model, the method calculates the TDF value. In section V we will explain usage of TDF.

A. Analysis of histories

Look at the history of Table II. We want to estimate how many times A occurred in [230000, 330000]. Table II shows that the occurrence time of concept A in 10th and 20th previous times are 260000 and 150000 respectively. 10 times occurrence of A in [150000, 260000] shows that this concept occurred three times in [230000, 260000]. So A occurred around 13 times in [230000, 330000].

V. Intelligent behavior

Having a concept shift, the correct decision making based on knowledge (here historical information) to behave proactively or reactively is an intelligent behavior. In next sections, first we will describe proactive and reactive behaviors in more detail and then we will explain how the method decides to behave either proactively or reactively.

A. Proactive behavior

Consider the current concept in data stream is A. We have some assumptions about what next concept will be e.g. the next concept is B or C or the other concepts. The method assigns a probability to each assumption and arranges them based on their probability in descending order. Without losing the order, assumptions with the probability more than a certain threshold will be inserted to a buffer. When concept changes, if method decides to behave proactively, this buffer is used to predict the next concept. First assumption in the buffer shows the most probable concept as the next one. If prediction is correct, it will be explicit that which previously learned data model is appropriate for new concept of data stream and details of this concept shift should be maintained in related histories. Having a wrong prediction, the method should decide to continue proactive behavior or not. If the method wants to continue proactive behavior, the next assumption in the buffer will be checked. When resuming proactive behavior is not rational anymore or buffer is empty, the method behaves reactively. We will explain decision making in details in section V.C. Now let’s answer to a question about assigning probability to assumptions.

How much is the probability of an assumption? Assume we

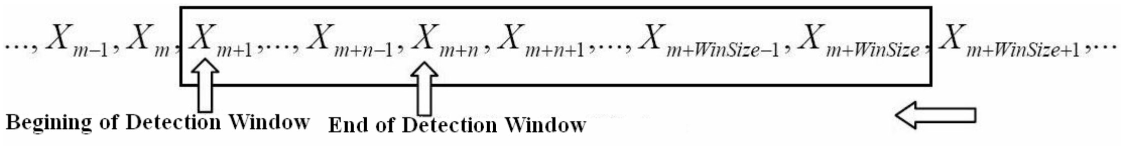
$\ldots, X_{m-1}, X_m, X_{m+1}, \ldots, X_{m+n-1}, X_{m+n}, X_{m+n+1}, \ldots, X_{m+WinSize-1}, X_{m+WinSize}, X_{m+WinSize+1}, \ldots$
Begining of Detection Window
End of Detection Window

Fig. 1 Monitoring Window and Detection Window

TABLE I HISTORY STRUCTURE FOR CONCEPTS AND CONCEPT SHIFTS

Counter	1	5	10	20	40	80	160	320	640	1280	First
147	620000	512000	445000	300000	200000	98000	-1	-1	-1	-1	800

TABLE II HISTORY STRUCTURE FOR CONCEPT A

Counter	1	5	10	20	40	80	160	320	640	1280	First
89	320000	290000	260000	150000	52000	8000	-1	-1	-1	-1	800

want to calculate the probability of B as the next concept (assume the current concept is A). As mentioned in section IV, the method maintains concept and concept shift histories. Using these histories, method determines how many times concept A and shift from A to B are occurred in (t_4,t_3), (t_3,t_2) and (t_2,t_1) when $t_4<t_3<t_2<t_1$ and t1 is the current time (In section IV.A we described how the we could determine how many times a concept or concept shift is occurred in a specific interval). Equation (2) is used to approximate the occurrence probability of B after A. n(A∩B) and n(A) in (2) show how many times AB and A are occurred in a specific interval. After estimation of the probabilities, we form (t_4, p_4), (t_3, p_3), (t_2, p_2) pairs while p_4, p_3 and p_2 are the probabilities in (t_4, t_3), (t_3, t_2) and (t_2, t_1) respectively. Now using an extrapolation method could help to estimate the value of p_1 in (t_1, p_1). p_1 shows the probability of assumption which introduces B as the next concept.

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{n(A \cap B)}{n(A)} \tag{2}$$

B. *Reactive behavior*

When a concept shift is detected, the method should replace the current data model with a proper one. If possible to predict the next concept, it replaces the current data model quickly; otherwise should wait to receive enough instances for training a stable data model. Please note that the DW size numbers of instances are sufficient to indicate that the current data model is not proper any more, but insufficient to induce what the proper model should be. After accumulation of enough instances, the method learns a new data model from scratch. If there is not a data model similar to the new learned one in memory, the method should create two new history (one to hold the information of new concept and the other for concept shift from previous concept to the new one) and store them along with the new data model in memory unless storing the information of concept shift in related histories will be sufficient. To check the existence of a data model similar to the new learned one, we should calculate the TDF feature of accumulated data. Having TDF we can calculate CF_3 and CF_4 using (3) and (4).

$$CF_3(i,j) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}[\sum x_i^2 - \frac{1}{n}(\sum x_i)^2] = \frac{1}{n}[CF_1(i,j) - \frac{1}{n}(CF_2(i,j))^2] \tag{3}$$

$$CF_4(i,j) = \frac{1}{n}(\sum x_i) = \frac{1}{n}(CF_2(i,j)) \tag{4}$$

So we define $CTDF_{current}$ as (5).

$$\begin{aligned} &((CF_3(1,1), CF_4(1,1), C_1), \ldots, (CF_3(k,1), CF_4(k,1), C_k)) \\ &\ldots \\ &((CF_3(1,m), CF_4(1,m), C_1), \ldots, (CF_3(k,m), CF_4(k,m), C_k)) \end{aligned} \tag{5}$$

We also calculate the CTDF of other data models in memory using their TDFes and then choose the *alpha* nearest ones to $CTDF_{current}$ based on Euclidean distance. Starting from data model of the nearest one, data models are checked by classification of accumulated data instances and comparing the result with true class labels. If number of misclassified instances is less than a certain threshold, the data model is similar to new learned one else the next data model will be checked.

C. *Proactive vs. Reactive*

When the method should behave proactively and when reactively? We propose that the method should behave proactively until it is rational. When resuming the proactive behavior is not rational, the method should switch to reactive behavior. How the method could distinguish the proactive behavior is not rational? As we said in section V.A, the assumptions, which their probability is more than a certain threshold, are inserted to a buffer in descending order. When a concept shift is detected, at the most, the method checks no more than the first *K* assumption in the buffer. *K* is determined by pseudo code of Fig. 2.

```
$\forall\ A_i \in \{A_1,...,A_m\} \to P(A_i \mid A_0) > Treshold$
$T_{train}$ = The required time to learn a data model from scratch
$T_{test}$ = The required time to test a data model on current data
D= $T_{train}$;
For (r=0; r < m ; r++){
    $D_{temp} = \sum_{i=1}^{r} (P(A_i \mid A_0) \times i \times T_{test}) + (1-\sum_{i=1}^{r} P(A_i \mid A_0)) \times (T_{train} + r \times T_{test})$
    if ($D > D_{temp}$) {   $D = D_{temp}$;
        K=i;    }
}
```

Fig. 2 Pseudo code to determine K value

In the above pseudo code, A_0 is the current concept and A_1 to A_m are concepts which their probability of being next concept are more than threshold and $P(A_1) > P(A_2) > \ldots > P(A_m)$. T_{train} and T_{test} show the required time to learn a data model from scratch and to test a previously learned data model on current data respectively. Assume the current concept in data stream is D and being the next concept probability for C, F, B and A are 0.35, 0.28, 0.17 and 0.03. We show D, C, F, B and A with A0, A1, A2, A3 and A4 respectively. Also T_{train} and T_{test} are 30 and 5. So we have:

r=0: required-time = 30

r=1:required-time=0.35× 1× 5+(1-0.35)× (30+1× 5)= 24.5

r=2:required-time=0.35× 1× 5+0.28× 2× 5+(1-0.35-0.28)× (30+2× 5)=19.35

r=3:required-time=0.35× 1× 5+0.28× 2× 5+0.17× 3 × 5+(1-0.35-0.28-0.17)× (30+3× 5)=16.1

r=4:required-time=0.35× 1× 5+...+(1-0.35-0.28-0.17-0.03)× (30+4× 5)=16.2

According to above expressions, the required time has minimum value when r is 3, so when a concept shift is detected, at the most, the method checks no more than first three assumptions (C, F and B) and switch to reactive behavior if the data models of these three assumptions, all are inappropriate for the new concept.

VI. Experiments

Experiments are conducted to evaluate the efficiency of the method for prediction in streaming data. First we introduce the datasets which are used and then we present results of the experiments.

A. Data

This paper 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. 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. So learning the length time of each concept and association among them will be very useful for real world applications.

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. Hence 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.

1) Artificial Data

Two benchmark stream data sets are employed that can cover all types of concept changes.

Stagger: This data set can simulate the scenario of concept shift [7], [8], [9]. Each instance consists of three attribute values: color {green, blue, red}, shape {triangle, circle, rectangle}, and size {small, medium, large}. There are three alternative underlying concepts:

If color = red and size = small, class = positive; otherwise class = negative.

If color = green and shape = circle, class = positive; otherwise class = negative.

If size = medium and large, class = positive; otherwise class = negative.

HyperPlane: We create synthetic data with drifting concepts using a moving hyperplane. A hyperplane in a d-dimensional space is represented by (6). Records satisfying (7) are labeled positive; otherwise, they are negative. A hyperplane has been used to simulate time-changing concepts because the orientation and the position of the hyperplane can be changed by changing the magnitude of the weight. We generate random records uniformly distributed in [0, 1]. Weights a_i are initialized by random values in [0, 1]. We set a_0 as (8) so that the hyperplane cuts the multidimensional space into two parts of the same volume. Thus, roughly half of the examples are positive, and the other half are negative.

$$\sum_{i=1}^{d} a_i x_i = a_0 \tag{6}$$

$$\sum_{i=1}^{d} a_i x_i < a_0 \tag{7}$$

$$a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i \tag{8}$$

2) Real World Data

We also used real life data set "nursery" in the University of California, Irvine (UCI) Machine Learning (ML) Repository. The data set has 10344 instances and eight dimensions. We randomly sample instances from the data set to generate a stream. To simulate concept shift we randomly select some attributes of the data set and change their values in a consistent way. One method is to shuffle the values; for example we can change (a_0, a_1, … , a_n) values as (a_1, a_2, … , a_n, a_0) for all instances and keep the class label intact.

B. *Results*

We selected WCE [3] and RePro [4] as rival methods and C4.5 decision tree algorithm as the learning method for all the three methods. Our evaluation measures are accuracy and the needed time to classify one million instances from data stream. We assume in starting point (time = 0) our method and RePro had been learned appropriate data models for commonly recurring concepts in data stream and afterward they will learn new data models for the concepts which never had been seen before. The monitoring window and detection window sizes are 500 and 200 respectively and the error threshold of detection window is 25%. After every 2500 instances, we change the concept of data to simulate concept shift.

Fig. 3 shows the performance of RePro and our method on one million data instance from HyperPlane in comparison with optimal method. In optimal mode we know what the next concept will be. As shown in this figure, our method performs closer to the optimal mode than RePro. Classification time of WCE is very big; therefore it is not included here. Fig. 4 and Fig. 5 show the results on Stagger and Nursery data streams. Table III shows the accuracy of the methods on data streams generated from HyperPlane, Stagger and Nursery data. WCE learns data models frequently, so it suit with gradual concept drifts and drop its accuracy when we have concept shift in data streams. In stopping mode when a concept shift is detected, our method pauses classification and fits its data model to the new concept but in non-stopping mode, the method continue classification and fits its data model in parallel with classification.

VII. Conclusion

Both foreseeing into the future and retrospection into the history have been proven important in many aspects of life. This paper incorporates those useful practices in mining data streams by using an intelligent behavior which is combination of proactive-reactive behaviors and has proposed a mechanism to organize into a concept history. The problem of intractable amount of streaming data is solved since data models are much more compact than raw data while still retain the essential information. Using histories to maintain the essential information and incorporating reactive and proactive behaviors, the proposed method was capable to achieve both effective and efficient predictions in various scenarios of concept change, including concept shift.

Acknowledgment

We would like to thank ITRC for their guidance and support in this research.

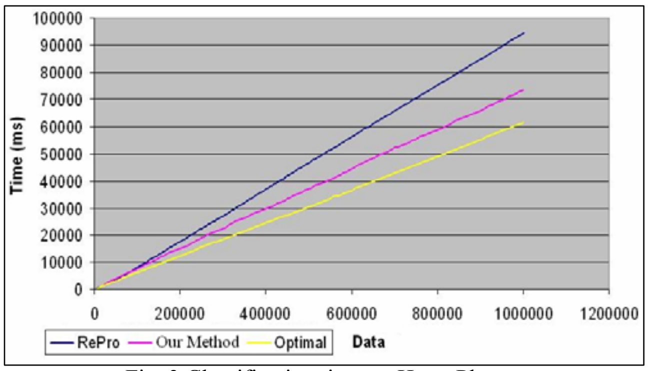

Fig. 3 Classification time on HyperPlane

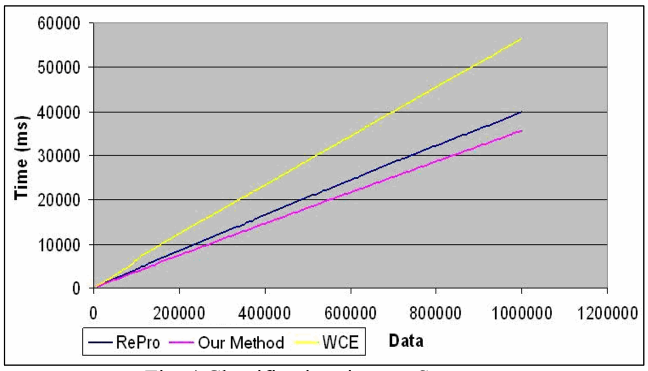

Fig. 4 Classification time on Stagger

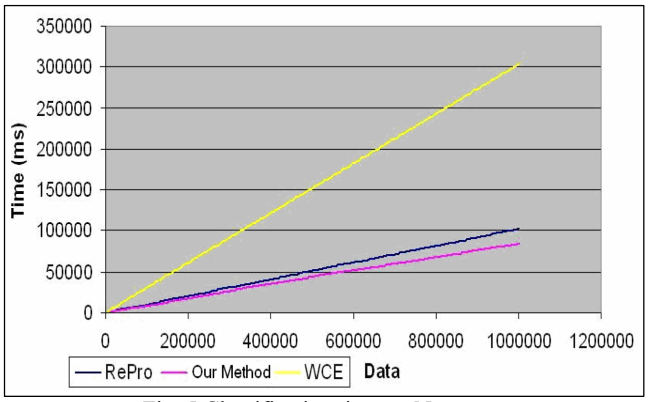

Fig. 5 Classification time on Nursery

Table III Classification Accuracy

Dataset	Mode	Our Method	RePro	W CE
HyperPlan	Non-Stopping	80.6	70.9	76.3
	Stopping	92.7	91.6	
Stagger	Non-Stopping	97.2	90.2	82.1
	Stopping	98.6	98.8	
Nursery	Non-Stopping	83.4	75.8	73.9
	Stopping	93.9	92.7	

REFERENCES

[1] P. Wang, H. Wang, X. Wu, W. Wang, and B. Shi, "A Low-Granularity Classifier for Data Streams with Concept Drifts and Biased Class Distribution", IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 9, Sept., 2007, pp. 1202-1213.

[2] P. Domingos, and G. Hulten, "Mining high-speed data streams", ACM Press, Boston, MA, 2000, pp. 71–80.

[3] 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, 2003, pp. 226–235.

[4] Y. Yang, X. Wu, and X. Zhu, "Mining in Anticipation for Concept Change: Proactive-Reactive Prediction in Data Streams", Journal of Data Mining and Knowledge Discovery, Springer, ISSN: 1384-5810, Volume 13, Number 3, November, 2006, pp. 261-289.

[5] G. Hulten, L. Spencer, and P. Domingos. "Mining time changing data streams". *In SIGKDD*, ACM Press, CA, 2001, pp. 97–106.

[6] W.N. Street and Y.S. Kim. "A streaming ensemble algorithm (SEA) for large-scale classification". In *SIGKDD*, 2001.

[7] E. Keogh, and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration", *In Proceedings of the 8thACMSIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 102–111.

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

[9] A. Tsymbal, "The problem of concept drift: Definitions and related work". Technical Report, Computer Science Department, Trinity College Dublin, 2004.

[10] X. Zhu, P. Zhang, X. Lin, Y. Shi, "Active Learning from Data Streams", IEEE International Conference on Data Mining, Omaha, Nebraska, USA, 2007.