Suppressing Model Overfitting in Mining Concept-Drifting Data Streams

Haixun Wang[†], Jian Yin[†], Jian Pei[‡], Philip S. Yu[†], and Jeffrey Xu Yu[††]
[†]IBM T. J. Watson Research, {haixun, jianyin, psyu}@us.ibm.com
[‡]Simon Fraser University, Canada, jpei@cs.sfu.ca
[††]The Chinese University of Hong Kong, yu@se.cuhk.edu.hk

ABSTRACT

Mining data streams of changing class distributions is important for real-time business decision support. The stream classifier must evolve to reflect the current class distribution. This poses a serious challenge. On the one hand, relying on historical data may increase the chances of learning obsolete models. On the other hand, learning only from the latest data may lead to biased classifiers, as the latest data is often an unrepresentative sample of the current class distribution. The problem is particularly acute in classifying rare events, when, for example, instances of the rare class do not even show up in the most recent training data. In this paper, we use a stochastic model to describe the concept shifting patterns and formulate this problem as an optimization one: from the historical and the current training data that we have observed, find the most-likely current distribution, and learn a classifier based on the most-likely distribution. We derive an analytic solution and approximate this solution with an efficient algorithm, which calibrates the influence of historical data carefully to create an accurate classifier. We evaluate our algorithm with both synthetic and real-world datasets. Our results show that our algorithm produces accurate and efficient classification.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*concept learning*; I.5.2 [**Pattern Recognition**]: Design Methodology—*classifier design and evaluation*

General Terms

Algorithms

Keywords

classifier, classifier ensemble, data streams, concept drift

1. INTRODUCTION

Mining data streams for actionable insights in real time has become an important and challenging task for a wide range of applications [7, 14, 16, 3, 12]. Compared to traditional data mining, mining data streams poses new challenges as data are streaming through instead of being statically available [3, 7, 14, 17]. As the underlying data generating mechanism is evolving over time, so are the data patterns that data mining systems intend to capture. This is known as *concept drifting* in the stream mining literature. To cope with concept drifts, stream mining systems must update their models continuously to track the changes. Moreover, in order to make time-critical decisions for stream ing data of huge volume and high speed, the stream mining systems must be efficient enough in updating the models.

There are some naïve approaches for handling streams with concept drifts. One is to incrementally maintain a classifier that tracks patterns in the recent training data, which is usually the data in the most recent sliding window. The other is to use the most recent data to evaluate classifiers learned from historical data and create an ensemble of "good" classifiers. As we will reveal in detail in this paper, both of the two approaches are subject to the same problem, namely, model overfitting, which has been known to affect the accuracy of a classifier.

Overfitting refers to the problem that models are too specific, or too sensitive to the particulars of the training dataset used to build the model. We argue that the following known issues that can lead to model overfitting have become more prevalent in the data streaming environment.

- **Insufficient training data.** In a streaming environment, it is essential to avoid having conflicting concepts in a training dataset. For this purpose, stream classifiers, such as the two approaches discussed above, enforce a constraint by learning models from data in a small window, for small windows are less likely to have conflicting concepts. However, a small window usually contains only a small number of training instances. Thus, the constraint makes the well-known cause of overfitting more prevalent.

- **Biased training data.** Stream data has the nature of being bursty. A large number of instances may arrive within a very short time, which seems to give us sufficient training data free of conflicting concepts. However, in many real-time applications, stream data that arrive within a short time interval tends to be concentrated in parts of the feature space. For example, a large amount of packets arrive in a bursty manner may all have the same source IP. Models learned from

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

or validated by such data will not generalize well for other data.

In mining static datasets, the problem of overfitting usually can be addressed by two approaches. First, we can enlarge the training dataset to reduce the risk of overfitting caused by insufficient training data. Second, we can use an evaluation data set to detect overfitting caused by biased training data – if a classifier's prediction accuracy relies on particular characteristics in the training data (e.g. the source IP address of the incoming packets), then the classifier's performance will be poor on an evaluation dataset as long as the evaluation dataset does not share these idiosyncrasies. Unfortunately, in the streaming environment, these methods are not applicable. When there are concept drifts, the enlarged part of the training dataset or the evaluation dataset may come from a different class distribution, which undermines our purpose of reducing overfitting.

In this paper, we propose a general framework that exploits concept drifting patterns to solve the model overfitting problem. It is generally impossible to capture concept drifts using a deterministic model because they happen unexpectedly. Using a stochastic model, we relate the current class distribution p with observations of the recent training data $D_n, D_{n-1}, \cdots$. Our problem of finding the most-likely current class distribution p is essientially the problem of finding the class distribution p that maximizes the probability of observing $D_n, D_{n-1}, \cdots$. Using standard optimization theory, we derive a solution for the most likely current class distribution. We then approximate this solution with an algorithm that combines the results of a set of classifiers trained over windows of historical training data. Our algorithm is very efficient – as concepts evolve over time, we only need to adjust the weights assigned to each of the historical classifiers.

2. OUR APPROACH

Given an ensemble of historical classifiers, we need to decide the weights of the classifiers in a meaningful way so that the ensemble can model the current class distribution. For simplicity, in our analysis, we assume that there are only two classes, positive and negative. Note that it is straightforward to generalize our analysis to multi-class cases.

2.1 Overview

Historical data should be leveraged to improve the estimation of the current class distribution. However, giving historical data the same weight as the current data hurts classification accuracy when there is a concept drift between the time the historical data is delivered and the current time. The task of this section is to derive a model that balances the influences of historical data so that we can derive a model that reflects the most likely current class distribution.

The timestamp of a historical dataset is an important piece of information to determine its influence to the current class distribution. The possibility of class distribution change increases monotonically as the length of time between the current time and the time when historical data is collected increases. However, it is generally impossible to model concept drift in a deterministic manner as concept drifts can happen at any point of the time between the current time and the time of the historical data. We use stochastic models to account for the uncertainty of the occurrences of concept drifts. Hidden Markov models (HMMs) are particularly promising because I) HMMs have been used in many fields to model uncertainty [15] and have demonstrated success in practice; II) HMMs allow us to decouple the observed outcome of training samples from the posterior class distribution. In a hidden Markov model, the states represent the posterior class distribution rather than observed training sample distributions. Training sample distributions follow the class distribution of the hidden state; III) HMMs allow us to make minimum assumptions as the stochastic process is embedded in the structure of the hidden Markov chain.

Another important piece of information is the density of historical data in different regions of the feature space. A classifier is less likely to be overfitted in regions where there are a large number of training records. It means predictions for samples in regions of high training data density should have a high weight. It follows that we should divide the feature space into regions and model each region with a Markov chain. In other words, a classifier will be weighted by both the time of its training data, and by their regions in the feature space.

Given the two pieces of information, our task is to find the most-likely current class distribution. It is tantamount to finding the current class distribution that maximizes the probability that we observe the data in the history. In this paper, we focus on the first issue, that is, what is the optimal way of weighting classifiers by time, and assume the feature space has already been partitioned. We leave the discussion on optimal feature space partition to a future paper.

The notations we use are summarized by Table 1.

2.2 Time and Space

First, we partition a stream into a sequence of windows, $W_1, W_2, \cdots, W_n$, of fixed time interval t, with W_n being the most recent window. Note that previous approaches partitioned the stream into windows of fixed number of instances [17]. We argue that the occurrence of concept drifts is more likely to be a function of time instead of the number of arriving instances. A bursty arrival of a large number of instances does not mean the underlying class distribution is changing at a fast rate. Thus, time windows are more natural in modeling concept drifts.

Second, we assume the feature space V has been partitioned into a set of non-overlapping regions, $S_1, S_2, \cdots, S_m$. The regions are aligned across all time windows. The trustworthiness of a classifier learned from data in a particular window may be different in different regions of the feature space. In the stream environment, a training dataset may have certain idiosyncrasies. For example, a burst of packets that arrive within a short time interval may all have the same source IP. The classifier learned from such a training data may have low authority in classifying records in other regions of the feature space. By giving different weights to classifiers in different regions, we can avoid overfitting caused by biased sampling.

In practice, there are many different ways to partition the feature space into multiple regions. We leave the discussion of optimal feature space partitioning to a future paper. To simplify our analysis, the partition method ensures that for records of the same region, a classifier always make the same prediction. For instance, in a decision tree, each leaf node in fact represents a region in the feature space. The class prediction for a test case that falls into a leaf node is $n_1/(n_1+n_2)$, where n_1 and n_2 are the number of positive and negative cases that belong to this leaf node in the training

t	size of time window
W_i	time window i
W_n	current time window
V	feature vector space
S_j	a region in the feature vector space, $V = \bigcup S_j$
N_i	total number of instances observed in time window W_i (of a given region)
f_i	observed class distribution in time window W_i (of a given region)
q_i	*posterior* class distribution in window W_i (of a given region)
q_n	*posterior* class distribution in the current window W_n (of a given region)
λ	the rate of concept drifts
C_i	the event that the latest concept drift occurs between time window W_{i-1} and W_i
$Y_i(x)$	the probability of the observation in time window W_i given that the class distribution is x
L_i	the probability of the observation across all time windows given C_i

Table 1: Notation

data. This means all cases that fall into the same leaf node will share the same prediction. We also align regions across all time windows. This is done by subdividing a region until it is contained in a certain leaf node of all classifiers.

For two-class data, the class distribution in any region can be sufficiently captured by a value in $[0, 1]$, which represents the probability that a test case in that region is positive. We use f_i to denote the positive class distribution in a region according to the classifier learned from data in W_i. In other words, f_i is the prediction given by the classifier to test cases that fall in the region. Given that there are N_i training cases in a region, we know there are $N_i f_i$ positive samples and $N_i - N_i f_i$ negative samples in the region.

2.3 Finding the most likely distribution

We leverage concept drift patterns to make a better use of historical data. To capture the non-deterministic nature of concept drifts in a region, we model the concept drift process as a continuous time Markov chain. Each state in the Markov chain represents a *posterior* class distribution at a particular point of time. The instances we observe at the time is a sample from the distribution. Concept drifts are modeled by change of states. A state can have multiple ingress edges. Assume for a given state there are m such edges representing transitions of rates $\lambda_1, \cdots, \lambda_m$ respectively. We use λ to denote the aggregated ingress transition rate, $\lambda = \sum_{i=1}^{m} \lambda_i$. An example is shown in Figure 1, where state A has three ingress edges with aggregated rate $\lambda = \lambda_1 + \lambda_2 + \lambda_3$.

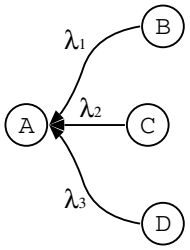

Figure 1: Markov Model

In the data stream environment, learning the structure of the Hidden Markov Model is tantamount to decoding the underlying data generation mechanism. This is itself a time-consuming task (to say the least), which makes it unrealistic for high volume, fast speed data streams. In our analysis, we assume the aggregate ingress rate of each state is the same. This actually means that the possibility of having concept drifts is distributed uniformly across the time axis. In other words, we assume that concept drifts are identically and independently distributed across the continuous time. Standard probability theory tells us that the only distribution satisfying this property is a Poisson process.

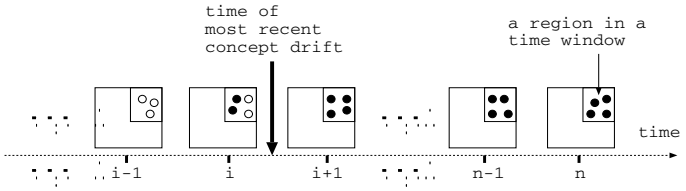

Figure 2: Concept drifts within a region across time windows

Figure 2 shows a region that is undergoing concept drifts across time. We model the most recent concept drift that has occurred. Let n be the timestamp of the current window. Let C_i represent the event that the most recent concept drift occurs between time i and time $i+1$. Given the aggregated rate of transition into the current state in the Markov model is λ, the probability that no concept drift occurs during an x time window internal is $1 - e^{-\lambda x t}$. Thus, the most recent state transition occurs between time window i and time window $i+1$ is

$$P(C_i) \quad = \quad e^{-\lambda(n-i)t} - e^{-\lambda(n-i+1)t} \tag{1}$$

Furthermore, if the *posterior* class distribution in the region at time i is x, the probability that we observe $N_i f_i$ positive instances out of the N_i total instances in the region is:

$$Y_i(x) = \binom{N_i}{N_i f_i} x^{N_i f_i} (1-x)^{(N_i - N_i f_i)} \tag{2}$$

Let q_i be the event that a random instance drawn from the region at time i is positive. Then, $P(q_i)$, the probability that a random instance is positive, is the positive class distribution at time i. If C_i is true, that is, no concept drift occurs after time $i+1$, we have $P(q_{i+1}|C_i) = P(q_{i+2}|C_i) = \cdots = P(q_n|C_i)$. Finally, given C_i, the probability that we observe the training samples across all the windows from

$W_{-\infty}$ to W_n is

$$L_i = \prod_{j=-\infty}^{n} Y_j(P(q_j|C_i)) = \prod_{j=-\infty}^{i} Y_j(P(q_j|C_i)) \times \prod_{j=i+1}^{n} Y_j(P(q_n|C_i)) \quad (3)$$

We assume that the states before the transition C_i are independent of the current state. This obviates the need for considering the structure of the whole Markov chain and allows us to focus our analysis on the current state instead. With this simplification, the first term $\prod_{j=-\infty}^{i} Y_j(P(q_j|C_j))$ in Eq 3 is a constant with respect to $P(q_n|C_i)$.

Based on the standard optimization theory, L_i is maximized when $\frac{dL_i}{dP(q_n|C_i)} = 0$ or at the boundary of $P(q_n|C_i)$, that is, when $P(q_n|C_i)$ equals 0 or 1. It is obvious that $L_i = 0$ when $P(q_n|C_i)$ equals 0 or 1 unless the training samples are either all positive or all negative. For all other cases, L_i is maximized when $\frac{dL_i}{dP(q_n|C_i)} = 0$. We have

$$\frac{dL_i}{dP(q_n|C_i)} = L_i \sum_{j=i}^{n} \left(\frac{N_j f_j}{P(q_n|C_i)} - \frac{N_j - N_j f_j}{1 - P(q_n|C_i)}\right) = 0$$

As $L_i \neq 0$, $\frac{dL_i}{dP(q_n|C_i)} = 0$ can occur only when

$$\sum_{j=i+1}^{n} \left(\frac{N_j f_j}{P(q_n|C_i)} - \frac{N_j - N_j f_j}{1 - P(q_n|C_i)}\right) = 0$$

Solve the equation for $P(q_n|C_i)$, we conclude that when

$$P(q_n|C_i) = \frac{\sum_{j=i+1}^{n} N_j f_j}{\sum_{j=i+1}^{n} N_j} \quad (4)$$

L_i in Eq 3 is maximized. In other words, given the observations in each window W_i and the assumption that the most recent concept drift occurs between time i and $i+1$, the most likely current class distribution is computed by Eq 4.

Since $P(\bigcup_i C_i) = 1$ and $C_i \bigcap C_{i'} = \emptyset$ when $i \neq i'$, we have

$$P(q_n) = \sum_i P(q_n|C_i)P(C_i)$$

Substituting $P(C_i)$ with Eq 1 , we get

$$P(q_n) = \sum_i \left(\frac{\sum_{j=i}^{n} N_j f_j}{\sum_{j=i}^{n} N_j}(e^{-\lambda(n-i)t} - e^{-\lambda(n-i+1)t})\right) \quad (5)$$

This shows that for any region, historical classifiers should be combined in the following manner. For class c, a classifier is weighted by the number of cases of class c in that region. In addition, its weight has an exponential time decay of parameter λ.

2.4 Algorithm

Our algorithm keeps k classifiers $\mathsf{C}_n, \cdots, \mathsf{C}_{n-k+1}$ trained from data in recent time windows. The user also provides parameter λ, the exponential decay rate. A larger λ discounts historical data more heavily and is used for streams of frequently changing class distributions.

We assume that the feature space has been partitioned into a set of non-overlapping regions. We leave the discussion of the optimal way of feature space partitioning to a future paper. Given a test case x, we consult the k classifiers.

Input: x: a test case
k: the total number of classifiers
λ: user defined exponential decay rate
Output: p: a probabilistic prediction of x

for *each classifier* $\mathsf{C}_i \in \{\mathsf{C}_n, \mathsf{C}_{n-1}, \cdots, \mathsf{C}_{n-k+1}\}$ **do**
  invoke $\mathsf{C}_i(x)$ to get the region S_i that contains x;
for *each region* S_i **do**
  let n_{i0} be the positive cases in region S_i;
  let n_{i1} be the negative cases in region S_i;
  $p_i = n_{i0}/(n_{i0} + n_{i1})$;
  $w_i = n_{i0}/(n_{i0} + n_{i1})(e^{-\lambda(n-i)t} - e^{-\lambda(n-i+1)t})$;
$p = \sum w_i p_i / \sum w_i$;
return p;

Algorithm 1: A region-based ensemble stream classifier

In addition to a probabilistic prediction for x, each classifier C_i returns the number of negative and positive training cases in the region where x falls into. Finally, we derive the probabilistic prediction and the weight based on the numbers.

Note that it is not necessary to consult every classifier. An improvement is to stop consulting classifiers back in the history once we are sure that they are unlikely to change final prediction (positive or negative). Since historical classifiers are heavily discounted, it can improve runtime performance. We omit detailed discussion here for lack of space.

3. EXPERIMENTS

We compare the performance of our approach, including time efficiency and classification accuracy, with previous approaches. The tests are conducted on a Linux machine with a 1.7 GHz CPU and 1 Gb main memory.

3.1 Data Sets

We create synthetic data whose concept drifting follows Poisson distribution. The class distribution is modeled by a hyperplane, which is denoted by the following equation in d-dimensional space.

$$\sum_{i=1}^{d} a_i x_i = a_0 \quad (6)$$

We label examples satisfying $\sum_{i=1}^{d} a_i x_i \geq a_0$ as positive, and examples satisfying $\sum_{i=1}^{d} a_i x_i < a_0$ as negative. Hyperplanes have been used to simulate time-changing concepts [14, 17]. However, previous approaches assume that the underlying concepts evolve in a smooth manner, while our testbeds simulate real life environment where abrupt changes can occur at any moment.

We generate random examples uniformly distributed in multi-dimensional space $[0, 1]^d$. Weights a_i $(1 \leq i \leq d)$ in Eq (6) are initialized randomly in the range of $[0, 1]$. We choose the value of a_0 so that the hyperplane cuts the multi-dimensional space in two parts of the same volume, that is, $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$. Thus, roughly half of the space is positive, and the other half negative. Noise is introduced by randomly switching the labels of $p\%$ of the examples. In our experiments, the noise level $p\%$ is set to 5%.

Number of records generated within a time unit follows a normal distribution with user provided mean and variance.

We study situations where data in each time unit may represent biased samples from the same class distribution T. However, it is not easy to quantify such a bias. In this study, we create biased class distributions in neighboring data units as a simulation. We generate a dataset wherein $s\%$ records are sampled from the positive class in T, and $1-s\%$ are sampled from the negative class in T. The next dataset will have $1-s\%$ positive records from T, and $s\%$ negative records from T. We denote such a stream as having a *sampling bias* $s\%$.

The rate of occurrence of concept drifting in the synthetic data is λ. To create such a change, we set certain weights a_i in Eq. 6 to random values within $[0, 1]$, which is tantamount to randomly moving the hyperplane to a new position. Furthermore, the users can also give a parameter k to specify the total number of dimensions whose weights are changing.

3.2 Performance Analysis

We study the time complexity of our stream classification algorithm and compare it with the *incrementally updated classifier* approach [14], and the *weight-by-accuray* approach [17]. We generate synthetic data streams where each data unit is of different average sizes. Note that both our approach and the *weight-by-accuray* approach train classifiers from each data unit. However, our approach does not apply historical classifiers on new training records in order to get weights for the classifiers.

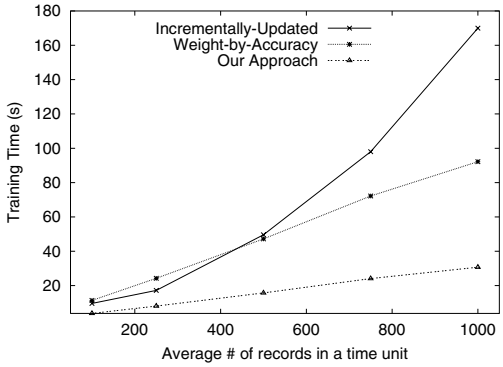

Figure 3: Training time and average partition size

Fig. 3 shows the time complexity of the three approaches. The tests are performed on a stream of 32 data units, and the X axis shows the average size of each partition. The incrementally updated classifier is trained from the entire 32 data units, and it is the most costly approach with regard to training time, as training a classifier (e.g., a C4.5 decisoin tree) is of superlinear complexity. Fig. 3 does not take into consideration the cost of model updating, which means in reality, the incrementally updated classifier is even more time consuming. The weight-by-accuray approach improves the single classifier by using the divide-and-conquer approach, however, it is still quite costly in comparison with our approach, which does not weight classifiers by their performance on the most recent training data.

From Fig. 3, it is clear that when the average partition size is small, all of the three approaches use less training time. Unfortunately, for ensemble-based approaches, using smaller partitions often results in lower classification quality. Fig. 4 shows this phenomena.

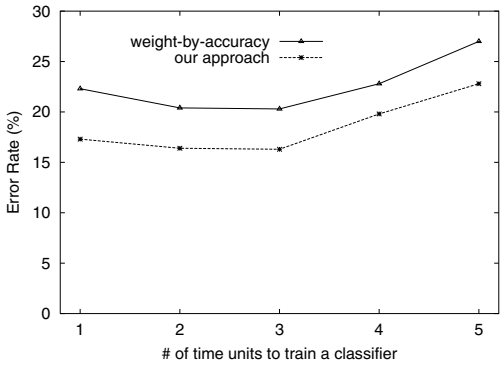

Figure 4: Error and average partition size

In Fig. 4, we see that the rate of classification error increases when partitions become larger. This is so because in a concept drifting environment, large partitions may contain conflicting concepts, and classifiers trained on a partition basis will have reduced quality. When a partition is small, the weight-by-accuray approach suffers because data in a small partition is often a biased sample from the true underlying class distribution. The experiment shown in Fig. 4 is conducted on synthetic data with a concept drifting rate of ($\lambda = 5$). We also introduce a 20% sampling bias in each unit data, which contains an average of 500 records.

Next, we study the change of data arrival rate on the accuracy of ensemble-based classifiers. Since a classifier is trained on data addrived during the same time unit, the variance in training data size may lead to some siginificant classification error. Furthermore, for the weight-by-accuray approach, the scarcity of the training data in the latest time unit may create erroneous weights for historical classifiers. In the experiment, we generate synthetic data with a concept drifting rate of ($\lambda = 5$), and we introduce a 20% sampling bias in each unit, which contains an average of 500 records. The weight-by-accuray approach uses an ensemble consisting 10 classifiers, and our approach uses a decay factor of 0.2. In Fig. 5, the X axis denotes the standard deviation in unit data size, and the Y axis denotes the rate of classification error. It indicates that our approach has advantage over the weight-by-accuray approach when deviation is high, although our approach uses much less training time.

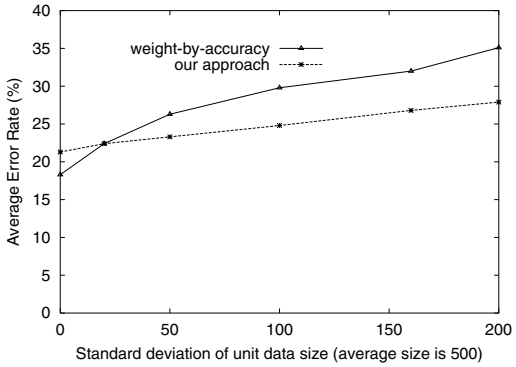

Figure 5: Error and deviation of unit data size

In Fig. 6, we study the effect of sampling bias on classification accuracy. The X axis represents the sampling bias, which ranges from 10% to 40%. The avarage size of each data unit is 500, and the standard deviation is 30. The other settings are the same as that of Fig. 5. The results indicate that data in each unit, which is very likely a to be biased sample from the true data distribution, may have a strong impact on the effectiveness of the weight-by-accuray approach. Our approach is able to reduce its effect.

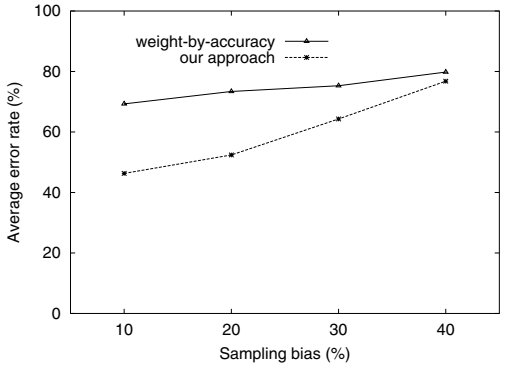

Figure 6: Error and sampling bias

4. RELATED WORK

Data stream processing has attracted much attention recently due to the explosive growth of many new classes of data stream applications. There is much work in this area including modeling, querying, and mining [1, 9, 11, 7, 14, 3, 12, 6, 13, 19, 18, 5, 4].

Traditional algorithms that require multiple scans of the training samples are inappropriate for our applications because our applications require real-time predications. Several incremental algorithm [10, 7, 14] refine models by continuously incorporating the influence of the new training samples and eliminating that of old ones. However, it is generally difficult to determine how fast the old data should be forgotten. Yang, et. al. [19] also proposed to use concept drifting patterns to improve classification accuracy. We model concept drifting differently and get different results.

There are also extensive studies on using classifier ensembles for traditional data mining including techniques such as Bagging [2], Boosting [8]. Unlike data stream mining, the model does not change in traditional data mining.

Our previous work [17] studies how to select a set of classifiers trained with data in previous time windows. In that work, we select and boost the previous classifiers based on their accuracy when they are applied to the training set in the current time window. It does not leverage concept drifting pattern and thus are not as accurate as the algorithm proposed in this paper.

5. CONCLUSION

This paper advocates exploiting concept drifting patterns to improve accuracy and efficiency of data stream classifiers. With stochastic models of concept drifting, we are able to formulate the classification problem as an optimization problem and derive a theoretical optimal solution. We then approximate the solution of the optimization problem by combining a set of traditional classifier. Our experimental results show that this approach results in significant improvement in terms of classification accuracy and efficiency compared to previous approaches that do not exploit concept drifting patterns.

6. REFERENCES

[1] B. Babcock, S. Babu, M. Datar, R. Motawani, and J. Widom. Models and issues in data stream systems. In *PODS*, 2002.

[2] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1-2):105–139, 1999.

[3] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *VLDB*, Hongkong, China, 2002.

[4] Yun Chi, Haixun Wang, Philip S. Yu, and Richard R. Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window data streams. In *ICDM*, 2004.

[5] Yun Chi, Philip S. Yu, Haixun Wang, and Richard Muntz. Loadstar: A load shedding scheme for classifying data streams. In *SIAM Data Mining*, 2005.

[6] Graham Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *SDM*, 2005.

[7] P. Domingos and G. Hulten. Mining high-speed data streams. In *SIGKDD*, pages 71–80, Boston, MA, 2000. ACM Press.

[8] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *ICML*, pages 148–156, 1996.

[9] L. Gao and X. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *SIGMOD*, Madison, Wisconsin, June 2002.

[10] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. BOAT– optimistic decision tree construction. In *SIGMOD*, 1999.

[11] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *SIGMOD*, pages 58–66, Santa Barbara, CA, May 2001.

[12] S. Guha, N. Milshra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *FOCS*, pages 359–366, 2000.

[13] Sudipto Guha and Boulos Harb. Wavelet synopsis for data streams: minimizing non-euclidean error. In *KDD*, pages 88–97, 2005.

[14] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *SIGKDD*, pages 97–106, San Francisco, CA, 2001. ACM Press.

[15] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[16] W. Nick Street and YongSeog Kim. A streaming ensemble algorithm (SEA) for large-scale classification. In *SIGKDD*, 2001.

[17] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *SIGKDD*, 2003.

[18] Peng Wang, Haixun Wang, Xiaochen Wu, Wei Wang, and Baile Shi. On reducing classifier granularity in mining concept-drifting data streams. In *ICDM*, 2005.

[19] Ying Yang, Xindong Wu, and Xingquan Zhu. Combining proactive and reactive predictions for data streams. In *SIGKDD*, pages 710–715, 2005.