

Active Learning from Data Streams*

Xingquan Zhu¹, Peng Zhang², Xiaodong Lin³, Yong Shi²

¹Dept. of Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA

²Graduate University, Chinese Academy of Sciences, Beijing 10080, China

³Dept. of Mathematical Sciences, University of Cincinnati, Cincinnati, OH 45221, USA

xqzhu@cse.fau.edu; {pzhang,yshi}@gucas.ac.cn; linxd@math.uc.edu

Abstract

In this paper, we address a new research problem on active learning from data streams where data volumes grow continuously and labeling all data is considered expensive and impractical. The objective is to label a small portion of stream data from which a model is derived to predict newly arrived instances as accurate as possible. In order to tackle the challenges raised by data streams' dynamic nature, we propose a classifier ensembling based active learning framework which selectively labels instances from data streams to build an accurate classifier. A *Minimal Variance* principle is introduced to guide instance labeling from data streams. In addition, a weight updating rule is derived to ensure that our instance labeling process can adaptively adjust to dynamic drifting concepts in the data. Experimental results on synthetic and real-world data demonstrate the performances of the proposed efforts in comparison with other simple approaches.

1. 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 [1-10]. Depending on data characteristics and data mining objectives, existing solutions in this area roughly fall into the following three categories: clustering and querying high speed data streams [3-4], association rule mining from stream data [5], and generating predictive models for continuous data streams [1-2, 6-10].

In the domain of classification, in order to generate a predictive model, it is essential to label a set of training examples from which a classifier can be trained. It is well accepted labeling training examples is a costly procedure [11], which requires comprehensive and intensive investigations on the instances, and incorrectly labeled examples will significantly deteriorate the model built from the data [20]. A common practice in addressing this problem is to use active learning techniques to selectively label a small number of instances from which an accurate predictive model can be formed. The main challenge of active learning is to identify those instances that should be labeled to achieve the highest prediction accuracy under the fact that one could not afford to label all instances. The *Uncertainty Sampling* (US) [12-13] principle has been employed extensively to achieve this goal. The intuition behind this concept is to label instances on which the current learner(s) has the highest uncertainty. A body of work has recently been proposed to address this

problem for static datasets, with an objective of building one single optimal model from the labeled data [14]. None of them fit in the setting of data streams.

For data streams with massive data volumes, the needs of employing active learning are compelling, simply because manually investigating and labeling all instances are infeasible for many applications. We believe that the challenges of active learning from data streams is threefold: (1) in data stream environments, the candidate pool is dynamically changing, whereas existing active learning algorithms are dealing with static datasets only; (2) the concepts, such as the genuine decision logics and class distributions, of the data streams are continuously evolving, whereas existing active learning algorithms only deal with constant concepts; and (3) because of its increasing data volumes, building one single model from all labeled data is computationally expensive for data streams, even if memory is not an issue, whereas most existing active learning algorithms rely on a model built from the whole collection of labeled data for labeling. In data stream environments, it is impractical to build one single learner from all previously labeled examples. On the other hand, as the concepts of the data streams evolve, aggregating all labeled instances may reduce the learner performances instead of adding help [2].

We present here our recent research efforts in addressing these challenges. In short, we propose a classifier ensembling based active learning framework, with an objective of maximizing the prediction accuracy of the classifier ensemble built from labeled stream data. This is achieved through the minimization of the classifier ensemble variance, which is used to guide our instance labeling.

2. Problem Statement and Simple Solutions

2.1 Problem Statement

A common solution in handling stream data is to partition the data into chunks, as shown in Figure 1. We can label a small portion of data in each chunk and build one classifier from labeled data. As a result, a set of classifiers are built and used to form a classifier ensemble to identify newly arrived data. Assume that stream data are partitioned chunk by chunk according to the user specified chunk size, assume further that once the algorithm moves to chunk S_n , all instances in previous data chunks, $\dots, S_{n-3}, S_{n-2}, S_{n-1}$, become inaccessible, except classifiers built from them (*i.e.*, $\dots, C_{n-3}, C_{n-2}, C_{n-1}$). Our objective here is labeling instances in data chunk S_n , such that a classifier C_n built from S_n , along with the most recent $k-1$ classifiers $C_{n-k+1}, C_{n-k}, \dots, C_{n-1}$ can form an accurate classifier ensemble (in terms of its prediction accuracy on unlabeled instances in S_n).

*This research has been supported by the National Science Foundation of China (NSFC) under Grant No.60674109.

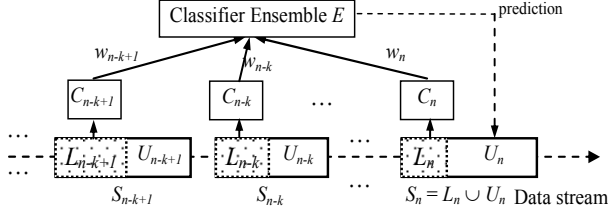


Figure 1: A general classifier ensemble framework for active learning from data streams

2.2 Simple Solutions

2.2.1 Random Sampling (RS)

Arguably, the simplest approach to solve our problem is random sampling, where instances in S_n are randomly sampled and labeled. Although simple, it turns out that RS performs surprisingly well in practice. The niche of random sampling stems from the fact that in data stream environments the class distributions may vary significantly across data chunks. While general active learning algorithms seek to label “important” instances, they may significantly change class distributions by favoring one class of instances, consequently, the labeled instances no longer reveal genuine class distributions in the data chunk. This problem is less severe for a static dataset where candidate pool is fixed and active learning algorithms are able to survey all candidates. Random sampling avoids this problem by randomly label instances. As a result, it can produce a training set with the most similar class distributions to the current data chunk.

2.2.2 Local Uncertainty Sampling (LU)

Another way of solving the problem is to disregard the dynamic nature of data streams and treat each data chunk S_n as a static dataset. Existing active learning algorithm can then be applied to S_n without considering any other data chunks. Because instance labeling is carried out independently in each data chunk, the weakness of LU is obvious: contributions of the labeled instances with respect to the global classifier ensemble are not clear, although each data chunk might indeed be able to label the most important instances locally.

2.2.3 Global Uncertainty Sampling (GU)

Global uncertainty sampling based active learning will use historical classifiers, along with the one from S_n , to form a classifier committee for instance labeling. Upon the receiving of a data chunk S_n , GU will randomly label a tiny set of instances from S_n and build a classifier C_n . This classifier along with $k-1$ historical classifiers $C_{n-k+1}, C_{n-k}, \dots, C_{n-l}$, will form a classifier committee, which is used to assess instances in S_n and label the ones with the largest uncertainty. The labeling process repeats until a certain number of instances in S_n are labeled. At any stage, the user may choose to rebuild C_n by using labeled examples in S_n to improve classifier committee’s capability in assessing remaining instances in S_n .

3. Classifier Ensemble Variance Reduction for Error Minimization

A Bayes optimal decision rule assigns input x to a class c_i if the a posteriori probability $p(c_i | x)$ is the largest among candidate classes. We assume that a given learner

(classifier)’s probability of classifying x to a class c_i deviates from $p(c_i | x)$ by an associated error $\varepsilon_i(x)$. The error can be decomposed into two terms: b_{c_i} represents the bias of the current learning algorithm and $\eta_{c_i}(x)$ is a random variable accounts for the variance. This gives Eq. (1) [2, 20-22].

$$f_{c_i}(x) = p(c_i | x) + b_{c_i} + \eta_{c_i}(x). \quad (1)$$

Since the same learning algorithm is used in our analysis, without loss of generality, we ignore the bias term [2, 20]. Consequently, the learner’s probability in classifying x into class c_i becomes

$$f_{c_i}(x) = p(c_i | x) + \eta_{c_i}(x). \quad (2)$$

When k base classifiers work together to form a classifier ensemble E , the probability of E in classifying an instance x is given by a linear combination of the probabilities produced by all its base classifiers. Here we employ a weighted classifier ensembling scheme, where each classifier C_n has a set of weight values w_i^n , $i=1, \dots, l$, that denotes C_n ’s weight with respect to class c_i . The probability of E in classifying x into class c_i is denoted by Eq. (3), where $f_{c_i}^m(x)$ denotes the probability of classifier C_m in classifying x into class c_i .

$$\begin{aligned} f_{c_i}^E(x) &= \frac{\sum_{m=n-k+1}^n w_i^m f_{c_i}^m(x)}{\sum_{m=n-k+1}^n w_i^m} \\ &= p(c_i | x) + \frac{\sum_{m=n-k+1}^n w_i^m \eta_{c_i}^m}{\sum_{m=n-k+1}^n w_i^m} \end{aligned} \quad (3)$$

This probability can be expressed as

$$f_{c_i}^E(x) = p(c_i | x) + \eta_{c_i}^E(x) \quad (4)$$

Where $\eta_{c_i}^E(x)$ is a random variable accounts for the variance of the classifier ensemble E with respect to class c_i , and

$$\eta_{c_i}^E = \frac{\sum_{m=n-k+1}^n w_i^m \eta_{c_i}^m}{\sum_{m=n-k+1}^n w_i^m} \quad (5)$$

Therefore, the variance of $\eta_{c_i}^E(x)$ is given by Eq. (6)

$$\sigma_{\eta_{c_i}^E}^2 = \frac{\sum_{m=n-k+1}^n \sum_{g=n-k+1}^n w_i^m w_i^g \text{cov}(\eta_{c_i}^m, \eta_{c_i}^g)}{\sum_{m=n-k+1}^n w_i^m \cdot \sum_{g=n-k+1}^n w_i^g} \quad (6)$$

Which can be rewritten as Eq. (7)

$$\sigma_{\eta_{c_i}^E}^2 = \frac{\sum_{m=n-k+1}^n (w_i^m)^2 \sigma_{\eta_{c_i}^m}^2 + \sum_{m=n-k+1}^n \sum_{g=n-k+1; g \neq m}^n w_i^m w_i^g \text{cov}(\eta_{c_i}^m, \eta_{c_i}^g)}{\left(\sum_{m=n-k+1}^n w_i^m \right)^2} \quad (7)$$

Assuming that $\eta_{c_i}^m$ and $\eta_{c_i}^g$ are independent for any classifier pairs C_g and C_m ($g \neq m$), the second term in Eq. (7) equals zero, thus the variance of $\eta_{c_i}^E(x)$ becomes Eq. (8), where

$\sigma_{\eta_{c_i}^m}^2$ denotes the variance of the random variable $\eta_{c_i}^m$. In our system, $\sigma_{\eta_{c_i}^m}^2$ is calculated by Eq. (9), where A_x is an evaluation set, $|A_x|$ denotes the number of instances in A_x ,

and $y_{c_i}^x$ is the genuine class probability of instance x . If x is labeled as class c_i , $y_{c_i}^x$ equals to 1, otherwise, it equals to 0.

$$\sigma_{\eta_{c_i}^E}^2 = \sum_{m=n-k+1}^n (w_i^m)^2 \sigma_{\eta_{c_i}^m}^2 / \left(\sum_{m=n-k+1}^n w_i^m \right)^2 \quad (8)$$

$$\sigma_{\eta_{c_i}^E}^2 = \frac{1}{|\Lambda_x|} \sum_{(x,c) \in \Lambda_x} (y_{c_i}^x - f_{c_i}^m(x))^2 \quad (9)$$

The total variance of the classifier ensemble E is then given by Eq. (10).

$$\sigma_{\eta^E}^2 = \sum_{i=1}^l \sigma_{\eta_{c_i}^E}^2 = \sum_{i=1}^l \left(\sum_{m=n-k+1}^n (w_i^m)^2 \sigma_{\eta_{c_i}^m}^2 / \left(\sum_{m=n-k+1}^n w_i^m \right)^2 \right) \quad (10)$$

According to Tumer et al's conclusion [15], a classifier's expected added error is proportional to its variance. Consequently, a classifier ensemble's expected error can be denoted by Eq. (11)

$$Err_{add}^E = \frac{\sigma_{\eta^E}^2}{S} \quad (11)$$

Eq. (11) states that in order to minimize a classifier ensemble error rate, we can minimize its variance instead, this can be achieved through the adjustment of the weight values w_i^n , $i=1, \dots, l$, associated with each of E 's base classifier C_n .

In order to minimize Eq. (10), we can compute its first partial derivative *w.r.t.* weight w_i^m and set it to zero, which will help us find weight values producing the smallest $\sigma_{\eta^E}^2$.

$$\begin{aligned} \frac{\partial \sigma_{\eta^E}^2}{\partial w_i^m} &= \frac{\partial \left[\sum_{j=1}^l \left(\left(\sum_{g=n-k+1}^n (w_j^g)^2 \cdot \sigma_{\eta_{c_j}^g}^2 \right) \cdot \left(\sum_{g=n-k+1}^n w_j^g \right)^{-2} \right) \right]}{\partial w_i^m} \quad (12) \\ &= 2 \frac{\sum_{g=n-k+1}^n w_i^g \left[w_i^m \sigma_{\eta_{c_i}^m}^2 - w_i^g \sigma_{\eta_{c_i}^g}^2 \right]}{\left(\sum_{g=n-k+1}^n w_i^g \right)^3} \end{aligned}$$

In order to solve Eq. (12), we adopt gradient descent rule [16], which minimizes $\sigma_{\eta^E}^2$ by iteratively updating w_i^m in inverse direction of $\sigma_{\eta^E}^2$'s first derivative. Therefore, the weight updating rule for w_i^m is given by Eq. (13).

$$w_i^m(K+1) = w_i^m(K) + \Delta w_i^m, \quad (13)$$

where K is the iteration of the weight updating process and Δw_i^m is given by Eq. (14)

$$\Delta w_i^m = -\xi \cdot \frac{\partial \sigma_{\eta^E}^2}{\partial w_i^m} \quad (14)$$

In Eq. (14), ξ is a parameter controlling the step size for Gradient Descent [16], in our experiments, we set $\xi=0.5$ which is a common default value in practice. Consequently, our weight updating rule becomes

$$w_i^m(K+1) = w_i^m(K) - \frac{\sum_{g=n-k+1}^n w_i^g(K) \cdot \left[w_i^m(K) \cdot \sigma_{\eta_{c_i}^m}^2 - w_i^g(K) \cdot \sigma_{\eta_{c_i}^g}^2 \right]}{\left(\sum_{g=n-k+1}^n w_i^g(K) \right)^3} \quad (15)$$

4. MV: Minimal Variance Based Active Learning from Data Streams

The main concern of active learning for data streams is to determine which instances should be labeled for each data chunk S_n , and our intuition is to label the ones which cause the current classifier ensemble to have the largest ensemble variance. We believe that those are the main instances responsible for classifier ensemble errors, and labeling them can reduce the overall classifier ensemble variance (hence error rates) on data chunk S_n . This intuition is the base for our *Minimal Variance* active learning algorithm for data streams (algorithm details are reported in Figure 2).

Assume that the algorithm has just finished data chunk S_{n-1} with the most recent $k-1$ classifiers denoted by $C_{n-k+1}, C_{n-k}, \dots, C_{n-1}$. In order to initiate active learning process on the new data chunk S_n , we first randomly label a tiny portion of instances from S_n and put them into L_n , followed by a learning process to build a classifier C_n from L_n . The k classifiers thus form a classifier ensemble E . After that, we initialize the weight values, w_i^m , $i=1, \dots, l$, for each classifier C_m , $m=n-k+1, \dots, n$, by using C_m 's prediction accuracy on L_n , as indicated on Step 6 of Figure 2. *I.e.*, we set $w_i^m(0) = w_j^m(0), \forall i, j \in [1, l]$, which is C_m 's accuracy on L_n .

After the above initialization process, for each unlabeled instance in U_n , we need to calculate its ensemble variance from current ensemble E , such that the one with the largest value will be selected for labeling. According to Eqs. (8) and (9), the variance of a classifier ensemble is based on its base classifiers' variance $\sigma_{\eta_{c_i}^m}^2$ on a specific evaluation set A_x . For

each unlabeled instance I_x in U_n , its evaluation set consists of all labeled instances in L_n as well as I_x itself, *e.g.*, $A_x = L_n \cup I_x$. Because the calculation of $\sigma_{\eta_{c_i}^m}^2$ requires that each instance in

A_x should have a class label, and I_x 's label is yet to be found, we will use E to assign a class label for I_x (which might be incorrect). We then use Eqs. (8) and (10) to calculate ensemble variance on A_x , this value is taken as the ensemble variance of I_x , as shown by Steps 8 (a) to 8(c).

After the calculation of the ensemble variance for all unlabeled instances in U_n , we label the one with the largest variance and add it to L_n . After that, we will update the weight associated with each base classifier to ensure that classifier ensemble E evolves towards a minimal variance on L_n , such that it can be beneficial for instance labeling in the next round. For this purpose, we recalculate each base classifier C_m 's variance $\sigma_{\eta_{c_i}^m}^2$ on L_n (the one calculated on Step 8(c) is not

accurate in the sense that the class of I_x is not labeled but predicted from E). Eq. (15) is used to calculate the new weight values for each base classifier C_m , $m=n-k+1, \dots, n$.

Following the weight updating process, the algorithm checks the following three conditions consecutively to ensure an iterative labeling process: (1) whether the user specified number of instances have been labeled (Step 12); (2) whether a new classifier C_n should be rebuilt after a certain number of instances are labeled (Step 13); (3) whether the algorithm should repeat and label next instance (Step 14).

Procedure: Active Learning from Data Streams**Given:** (1) current data chunk S_n ; and (2) $k-1$ classifiers $C_{n-k+1}, \dots, C_m, \dots, C_{n-1}$ built from the most recent data chunks;**Parameters:** (1) α , the percentage of instances should be labeled from S_n ; (2) e , # of epochs in labeling α percentage of instances from S_n ;**Objective:** Label α of instances in S_n , such that the classifier built from L_n , denoted by C_n , along with the previous $k-1$ classifier can form a classifier ensemble as accurate as possible (in terms of its accuracy on unlabeled instances in S_n).

1. $L_n \leftarrow \emptyset$; $U_n \leftarrow S_n$
2. $L_n \leftarrow$ Randomly label a tiny portion, e.g. 1~2.5%, of instances from S_n .
3. $\kappa \leftarrow 0$ // recording the total number of labeled instances
4. Build a classifier C_n from L_n
5. $K \leftarrow 0$ // recording the number of instances labeled in the current epoch
6. Initialize weight values for each classifier $C_m, m=n-k+1, \dots, n$, where $w_j^m(0) = w_j^m(0), \forall i, j \in [1, l]$, which is C_m 's prediction accuracy on L_n .
7. Use $C_{n-k+1}, \dots, C_m, \dots, C_n$ to form a classifier ensemble E .
8. **For** each instance I_x in U_n
 - a. Use ensemble E to predict a class label for I_x .
 - b. Build an evaluation set $A_x = L_n \cup \hat{I}_x$, where \hat{I}_x means I_x with a predicted class label.
 - c. Calculate each classifier C_m 's variance on A_x (Eq. (11)), and feed the value into Eq. (8) to calculate ensemble variance. Calculate total variance over all classes by Eq. (10) and use this value as I_x 's expected ensemble variance.
9. **EndFor**
10. Choose instance I_x in U_n with the largest ensemble variance, label I_x , and put it into L_n , i.e., $L_n = L_n \cup I_x$; $U_n = U_n \setminus I_x$
11. Recalculate the variance of each base classifier C_m on L_n , and use Eq. (15) to update weight values for each classifier c_m .
 - a. $w_i^m(K+1) = w_i^m(K) + \Delta w_i^m$; $i=1, \dots, l$
12. $\kappa \leftarrow \kappa + 1$; $K \leftarrow K + 1$
13. **If** $\kappa \geq |S_n| \cdot \alpha$ Exit // exist if α percentage of instance are labeled
14. **Else if** $K \geq (|S_n| \cdot \alpha / e)$ Repeat step 4 // rebuild model C_n if one epoch ends
15. **Else** Repeat step 8 //continue instance labeling without rebuilding C_n

Figure 2: Active learning for data streams

5. Experimental Comparisons

5.1 Experimental Setting

5.1.1 General Setting

In order to compare different active learning methods and justify the quality of instances labeled by them, we build a classifier ensemble E for each of them by using the same framework in Figure 1. Therefore, if one classifier ensemble outperforms others, we can conclude that this is due to the fact that the instances used to train the classifier ensemble are of better quality. Notice that different active learning methods will select different portions of instances from S_n , this leads to different test sets for classifier ensembles. For comparison purposes, we have to make sure that all methods are compared based on the same test sets. Therefore, our comparisons are made by comparing ensemble classifiers on the average prediction accuracies on all instances in data chunk S_n over the whole stream.

All methods are implemented in Java with an integration of WEKA data mining tool [17]. All tests are conducted on a PC machine with a 2.0G CPU and 512MB memory.

5.1.2 Data Streams

Synthetic data: We employ a hyper-plane based synthetic data stream generator, due to its convenience in controlling the magnitude of the concept drifting and its popularity in stream data mining research [2, 6-7]. The hyper-plane of the data generation is controlled by the function in Eq. (16).

$$f(x) = \sum_{i=1}^d \frac{a_i}{x_i + (x_i)^2} \quad (16)$$

In Eq. (16), d is the total dimensions of the input data x . Each dimension $x_i, i=1, \dots, d$, is a value randomly generated in the range of $[0, 1]$. A weight value $a_i, i=1, \dots, d$, is associated with

each input dimension, and the value of a_i is initialized randomly in the range of $[0, 1]$ at the beginning. In data generation process, we will gradually change the value of a_i to simulate the concept drifting. To generate a class label for each instance x , we first generate a threshold value $a_0 = \sum_{i=1}^d a_i / (0.5 + 0.5^2)$, then we set a region $[\mu a_0, \nu a_0]$ for each class. If $f(x)$ falls into the region of one specific class, we will use this class to label x .

The general concept drifting is simulated and controlled through the following three parameters [2, 7-8]: (1) t , controlling the magnitude of the concept drifting (in every N instances); (2) p , controlling the number of attributes whose weights are involved in the change; (3) h and $n_i \in \{-1, 1\}$, controlling the weight adjustment direction for attributes involved in the change. In summary, we use c2-1100k-d10-p5-N1000-t0.1-h0.2 to denote a two class data stream with 100k instances, each containing 10 dimensions. The concept drifting involves 5 attributes, and attribute weights change with a magnitude of 0.1 in every 1000 instances and weight adjustment inverts the direction with 20% of chance.

Real-world Data: Due to the unavailability of public benchmark data streams (from classification perspectives), we select three relatively large datasets from UCI data repository [18] and treat them as data streams for active learning. The datasets we selected are Adult (two classes and 48,842 instances), Covtype (7 classes and 581,012 instances), and Letter (26 classes and 20,000 instances).

5.1.3 Benchmark Methods

For comparison purposes, we implemented three simple methods introduced in Section 2: random sampling (RS), local uncertainty (LU), and global uncertainty (GU) sampling based active learning. Our minimal variance based active learning method is denoted by MV.

5.2 Experimental Results

5.2.1 Active Learning with a Fixed α Value

We use C4.5 [19] as our base learner and apply active learning to three types of synthetic data streams: two-class (Table 1.1) three-class (Table 1.2) and four-class (Table 1.3). For each data stream, we report its results of using different chunk sizes (varying from 250 to 2000). We fix α value to 0.1 and set value k to 10, which means that only the most recent 10 classifiers are used to form a classifier ensemble.

Comparing all four methods, we can easily conclude that MV receives the best performances across all data streams. The local uncertainty based method (LU) is not an option for active learning from data streams, and its performances are constantly worse than RS regardless of whether the data streams are binary or multi-class. Although GU outperforms random sampling (RS) quite often, for multi-class data streams (e.g. four classes) its performances are unsatisfactory and are almost always inferior to RS. The results from RS are surprisingly good, and it is generally quite difficult to beat RS with a substantial amount of improvement. As we have analyzed in Section 2, random sampling naturally address the challenges of varying class distributions in data streams by randomly label instances in the data chunk. As a result, it produces a subset with the most similar class distributions to the genuine distributions in the data chunk.

In order to compare different methods at individual data chunk level, we record each method’s accuracy on each single data chunk and report the results (10 times average) in Figures 3 (a) to 3(c), where the x -axis represents data chunk ID in its temporal order and the y -axis shows the classifier ensemble accuracy. The significant accuracy increase from data chunks 1 to 11 in Figures 3(a) to 3(c) is caused by the increasing number of base classifiers at the beginning of the data streams. The results in Figures 3(a) to 3(c) indicate that the accuracies of the data chunks fluctuate frequently and the fluctuations are quite significant for some data streams (e.g., Figure 3(b)). MV consistently outperforms other methods across all data chunks. This observation asserts that for concept drifting data streams with constant changing class distributions and continuous evolving decision surfaces, MV can adaptively label instances from data chunks to build a superior classifier ensemble. The advantage of MV can be observed across different types of data streams (binary-class and multi-class), as well as data chunks inside each data stream.

5.2.2 Active Learning with Different α Values

In Figure 4, we compare all four methods for different α values. Overall, MV and GU achieve the best performances,

and LU performs inferior to RS in the majority of cases. This, again, asserts that applying traditional uncertainty sampling locally to each single data chunk is inappropriate for data streams, where dynamically changing class distributions and evolving concepts require an active learning algorithm to take these issues into consideration for effective instance labeling.

As we discussed in Section 2, GU extends Query by Committee to data streams by taking classifiers learnt from different data chunks as committee members. In the original QBC, the committee members are learned from the data with the same distributions (randomly sampled from the labeled data), therefore committee members are considered similar to each other with relatively small variances in their predictions. In data stream environments, in addition to the fact that the committee classifiers are learned from different portion of stream data, the concept drifting in the data chunks also render classifiers significantly different from each others. As a result, weighted average uncertainty over all committee members no longer reveals the genuine uncertainty of the classifier ensemble formed by them. The results in Figure 4 support our hypothesis very well. As we can see, MV constantly outperforms GU across all α values. For multi-class data streams, the performances of GU are unsatisfactory and are inferior to random sampling sometime.

Table 1.1: Accuracy on c2-I50k-d10-p5-N1000-t0.1-h0.2 ($\alpha=0.1$)

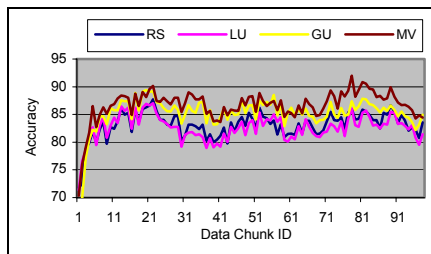
Chunk Size	RS	LU	GU	MV
250	74.54 \pm 2.89	73.70 \pm 2.75	75.51 \pm 3.35	81.08 \pm 2.85
500	83.07 \pm 2.42	82.56 \pm 2.36	85.16 \pm 2.95	86.79 \pm 2.51
750	86.10 \pm 2.75	85.69 \pm 3.13	88.14 \pm 3.04	88.67 \pm 2.74
1000	86.43 \pm 3.52	86.11 \pm 4.12	88.79 \pm 3.47	89.09 \pm 3.39
2000	86.47 \pm 5.01	85.94 \pm 4.82	88.69 \pm 4.27	88.88 \pm 4.27

Table 1.2: Accuracy on c3-I50k-d10-p5-N1000-t0.1-h0.2 ($\alpha=0.1$)

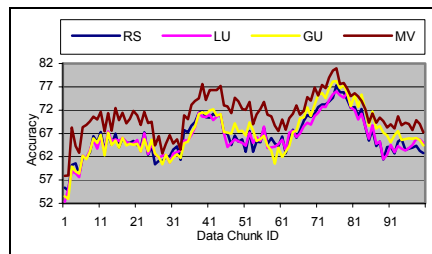
Chunk Size	RS	LU	GU	MV
250	56.36 \pm 2.71	55.46 \pm 2.64	56.57 \pm 2.71	64.41 \pm 3.87
500	66.31 \pm 4.30	66.11 \pm 4.23	66.99 \pm 4.98	71.59 \pm 4.23
750	71.79 \pm 2.86	70.49 \pm 3.61	72.59 \pm 3.77	74.39 \pm 3.99
1000	75.29 \pm 2.88	74.23 \pm 3.27	76.45 \pm 3.72	78.22 \pm 4.38
2000	73.92 \pm 6.11	72.97 \pm 6.01	76.37 \pm 4.64	75.98 \pm 5.44

Table 1.3: Accuracy on c4-I50k-d10-p5-N1000-t0.1-h0.2 ($\alpha=0.1$)

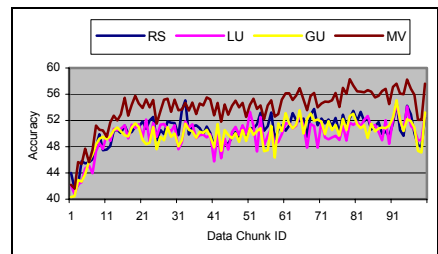
Chunk Size	RS	LU	GU	MV
250	42.27 \pm 2.01	41.38 \pm 1.80	41.34 \pm 2.09	46.45 \pm 2.04
500	50.47 \pm 2.34	49.81 \pm 2.47	49.91 \pm 2.45	53.95 \pm 3.04
750	58.11 \pm 3.65	56.75 \pm 3.26	56.48 \pm 3.09	59.86 \pm 3.88
1000	63.08 \pm 3.67	62.72 \pm 4.07	61.92 \pm 3.65	64.04 \pm 4.11
2000	71.87 \pm 4.47	70.67 \pm 4.98	70.98 \pm 5.09	72.13 \pm 5.33



(a) c2-I50k-d10-p5-N1000-t0.1-h0.2



(b) c3-I50k-d10-p5-N1000-t0.1-h0.2



(c) c4-I50k-d10-p5-N1000-t0.1-h0.2

Figure 3: Classifier ensemble accuracy across different data chunks: (a) two-class; (b) three-class; (c) four-class data streams ($\alpha=0.1$, $e=3$)

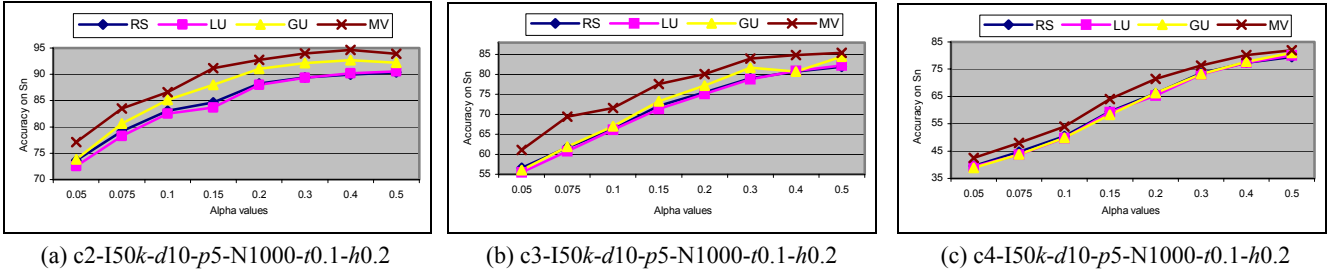


Figure 4: Classifier ensemble accuracy with respect to different α values (chunk size 500, $e=3$)

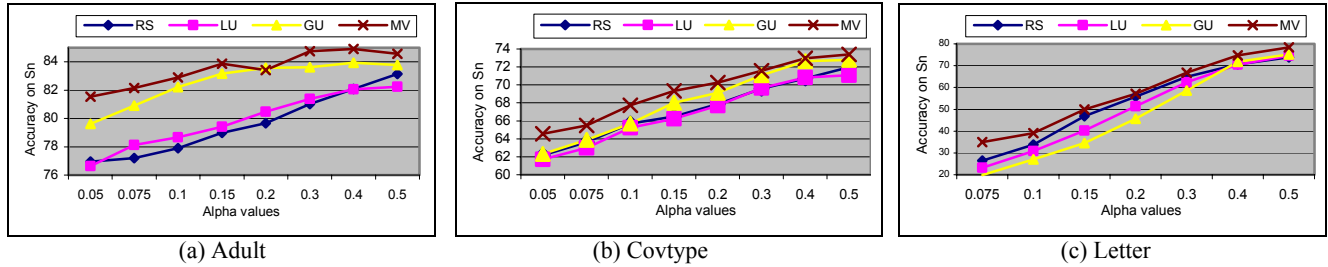


Figure 5: Classifier ensemble accuracy on data chunk S_n (chunk size 500, $e=3$)

5.2.3 Active Learning from Real-world Data

In Figure 5 we report the algorithm performances on three real-world data. Different from synthetic data streams where data chunks have strong correlations, the data chunks of the real-world data do not share such property, and the concept drifting among data chunks are not clear to us (in fact, we do not even know the genuine concepts of the data). The results in Figure 5 assert that MV consistently outperforms other methods. Although GU is able to perform well for both Adult and Covtype, its performance on Letter is significantly worse than all the other methods, where the accuracy of GU can be as much as 20% lower than MV. Considering Letter is a sparse dataset with 26 evenly distributed classes, it suggests that for data streams with a large number of classes, traditional uncertainty sampling would perform poorly. This asserts our analysis in Section 2 and Section 5.2, that for sparse data streams with a large number of classes, the classifiers built from different data chunk vary significantly. Simply calculating the averaged uncertainty over all committee classifiers (like QBC does) is not a good solution, and can produce much worse results than random sampling.

6. Conclusions

In this paper, we proposed a new research topic on active learning from data streams, where data volumes continuously increase and data concepts dynamically evolve, and our objective is to label a small portion of stream data to form a classifier ensemble to accurately predict newly arrival instances. In order to address the problem, we proposed a Minimal Variance (MV) principle, where the key is to label instances which cause the classifier ensemble to generate the largest variance. We derived a weight updating rule to ensure that labeled instances can help classifier ensemble to adapt to dynamic data streams and evolve towards the global minimal variance, and consequently minimal error rates. We have also provided a framework which accommodates the above minimal variance principle for active learning from data streams. Our experimental results on synthetic and real-world data demonstrated the effectiveness of the proposed design.

References

1. P. Domingos & G. Hulten, Mining high-speed data streams, Proc. of KDD, 2000.
2. H. Wang, W. Fan, P. Yu, & J. Han, Mining concept-drifting data streams using ensemble classifiers, Proc. of KDD, 2003.
3. C. Koch, S. Scherzinger, N. Schweikardt, & B. Stegmaier, Schema-based scheduling of event processors and buffer minimization for queries on structured data streams. Proceedings of VLDB 2004.
4. S. Guha, N. Milshra, R. Motwani, & L. O'Callaghan, Clustering data streams, Proc. of FOCS, 2000.
5. Y. Chi, H. Wang, P. Yu, & R. Muntz, Moment: Maintaining closed frequent itemsets over a stream sliding window data streams, Proc. of ICDM 2004.
6. H. Wang, J. Yin, J. Pei, P. Yu, & J. Yu, Suppressing model overfitting in mining concept-drifting data streams, Proc. of KDD, 2006.
7. Y. Yang, X. Wu, & X. Zhu, Combining proactive and reactive predictions of data streams, Proc. of KDD, 2005.
8. W. Street & Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, Proc. of KDD, 2001.
9. W. Fan, Y. Huang, P. Yu, Decision tree evolution using limited number of labeled data items from drifting data streams, Proc. of ICDM 2004.
10. W. Fan, Y. Huang, H. Wang, & P. Yu, Active mining of data streams, Proc. of SDM 2004.
11. D. Cohn, L. Atlas, R. Ladner, Improving Generalization with Active Learning, *Machine Learning* 15(2), 1994.
12. H. Seung, M. Opper, & H. Sompolinsky, Query by committee, Proc. of COLT, 1992.
13. Y. Freund, H. Seung, & E. Tishby, Selective sampling using the query by committee algorithm, *Machine Learning*, 1997.
14. S. Hoi, R. Jin, J. Zhu, & M. Lyu, Batch mode active learning and its application to medical image classification, Proc. of ICML, 2006.
15. K. Tumer & J. Ghosh, Error correlation and error reduction in ensemble classifier, *Connection Science*, 8(3-4), 1996.
16. S. Haykin, *Neural Networks: A comprehensive foundation*, Prentice Hall, 1998.
17. I. Witten & E. Frank, *Data mining: practical machine learning tools and techniques*, Morgan Kaufmann, 2005.
18. D. Newman, S. Hettich, C. Blake, C. Merz. UCI Repository of machine learning, 1998.
19. J. Quinlan, *C4.5: Programs for Machine learning*, M. Kaufmann, 1993.
20. Zhu X., & Wu X., Class Noise vs. Attribute Noise: A Quantitative study of their impacts, *Artificial Intelligence Review*, 22, 2004.