A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions *

Jing Gao† Wei Fan‡ Jiawei Han† Philip S. Yu‡
†University of Illinois at Urbana-Champaign
‡IBM T. J. Watson Research Center
†{jinggao3@uiuc.edu, hanj@cs.uiuc.edu} ‡{weifan,psyu}@us.ibm.com

Abstract

In recent years, there have been some interesting studies on predictive modeling in data streams. However, most such studies assume relatively balanced and stable data streams but cannot handle well rather skewed (e.g., few positives but lots of negatives) and stochastic distributions, which are typical in many data stream applications. In this paper, we propose a new approach to mine data streams by estimating reliable posterior probabilities using an ensemble of models to match the distribution over under-samples of negatives and repeated samples of positives. We formally show some interesting and important properties of the proposed framework, e.g., reliability of estimated probabilities on skewed positive class, accuracy of estimated probabilities, efficiency and scalability. Experiments are performed on several synthetic as well as real-world datasets with skewed distributions, and they demonstrate that our framework has substantial advantages over existing approaches in estimation reliability and predication accuracy.

1 Introduction

Many real applications, such as network traffic monitoring, credit card fraud detection, and web click stream, generate continuously arriving data, known as data streams [3]. Since classification could help decision making by predicting class labels for given data based on past records, classification on stream data has been extensively studied in recent years, with many interesting algorithms developed [10, 14, 7, 2]. However, there are still some open problems in stream classification as illustrated below.

First, descriptive (non-parametric) and generative (parametric) methods are two major categories for stream classification algorithms. Existing algorithms simply choose one from them, and have not explained why descriptive model is chosen or vice versa. Second, many stream classification algorithms focus on mining concept-drifting data streams, however, they recognize concept drift as change in $\mathcal{P}(y|\mathbf{x})$, the conditional probability of class y given feature vector $\mathbf{x}$. In reality, however, we could only observe the changes in the joint probability $\mathcal{P}(\mathbf{x}, y)$ and it is hard to tell whether the changes are caused by changes in $\mathcal{P}(\mathbf{x})$ or $\mathcal{P}(y|\mathbf{x})$. Third, some algorithms would predict a class label for a given test example. This may work well for deterministic problems, yet is not reasonable for a stochastic application or previously deterministic problem unknowingly evolving into a stochastic one. Compared with deterministic problems, where each example strictly belongs to one class, stochastic processes assign labels to test examples based on some probability distribution. In this scenario, a model that could generate accurate probability estimates is preferred.

Another important issue is that existing stream classification algorithms typically evaluate their performances on data streams with balanced class distribution. It is known that many inductive learning methods that have good performances on balanced data would perform poorly on skewed data sets. In fact, skewed distribution can be seen in many data stream applications. In these cases, the positive instances are much less popular than negative instances. For example, the online credit card fraud rate of US is just 2% in 2006. On the other hand, the loss functions associated with classes are also unbalanced. The cost of misclassifying a credit card fraud as normal will impose thousands of dollars loss on the bank. The deficiency in inductive learning methods on skewed data has been addressed by many people [15, 5, 4]. Inductive learner's goal is to minimize classification error rate, therefore, it completely ignores the small number of positive examples and predicts every example as negative. This is definitely undesirable.

In light of these challenges, we first provide a sys-

*The work was supported in part by the U.S. National Science Foundation NSF IIS-05-13678/06-42771 and NSF BDI-05-15813.

tematic analysis on stream classification problems in general. We formally define four kinds of concept changes in stream data, and show that an effective method should work equally well in each of the four cases. Also, we argue that a descriptive model that could approximate posterior probability $\mathcal{P}(y|\mathbf{x})$ would be most desirable for real stream classification problems. Although these observations are applicable to different scenarios, we are particularly interested in skewed stream mining problem since it is an important problem and no existing methods can handle them well. The main contributions in this paper are as follows:

1. The concept drift in data streams is formally defined and analyzed. We show that the expected error rate is not directly related to concept drift and could be reduced by training a model on the most up-to-date data.

2. We analyze several important stream classification problems systematically. By comparing descriptive models with generative models, label prediction with probability estimation, we draw the conclusion that in concept-drifting stream mining, a descriptive model that could generate high quality probability estimates is the best choice.

3. We propose an effective and efficient algorithm to classify data streams with skewed class distribution. We employ both sampling and ensemble techniques in the algorithm and show their strengths theoretically and experimentally. The results clearly indicate that our proposed method generates reliable probability estimates and significantly reduces the classification error on the minority class.

The rest of the paper is organized as follows. Section 2 analyzes different kinds of concept drifts and discusses important problems in stream classification. In Section 3, we introduce an ensemble approach for mining skewed data streams and demonstrate its advantages through theoretical analysis. Experimental results on the ensemble approach are given in Section 4. Finally, related work is presented in Section 5, followed by conclusions in Section 6.

2 Inductive Learning on Concept-Drifting Data Streams

In this section, we point out three unattended problems in stream classification and provide a thorough analysis for these problems, namely, the possible concept changes, the model of inductive learning and the posterior probability estimation.

2.1 Concept-Drifting Data Streams In this section, we describe different kinds of concept changes and discuss how error rate changes as the result of concept-drifts. Assuming that the true probability distribution $\mathcal{P}(\mathbf{x}, y) = \mathcal{P}(y|\mathbf{x}) \cdot \mathcal{P}(\mathbf{x})$ is given, then the expected error by a model is $Err = \int_{(\mathbf{x},y)\in\mathcal{P}(\mathbf{x},y)} \mathcal{P}(\mathbf{x})(1 - \mathcal{P}(y_p|\mathbf{x}))\mathrm{d}\mathbf{x}$ where y_p is the predicted class label and is chosen to be equal to $\mathrm{argmax}_y \mathcal{P}(y|\mathbf{x}, \theta)$ under 0-1 loss. Note that $\mathcal{P}(y|\mathbf{x}, \theta)$ refers to the estimated probability by a model θ, and it can be different from the true conditional probability $\mathcal{P}(y|\mathbf{x})$. Let $y_M = \mathrm{argmax}_y \mathcal{P}(y|\mathbf{x})$, then those examples with $y_p \neq y_M$ are "rooms" for improvement. Concept-drift is best described as changes in $\mathcal{P}(\mathbf{x}, y)$. Since it is composed of feature probability $\mathcal{P}(\mathbf{x})$ and class label conditional probability $\mathcal{P}(y|\mathbf{x})$, the change of the joint probability can be better understood via the changes in either of these two components. There are four possibilities, as discussed below. In each case, we demonstrate that building a new model on the most recent data could help reduce Err. We define 'the most recent data' as the data held in memory at current time among continuously arriving stream data. It is optimal to always update the model according to the most recent data no matter how concepts evolve.

No Change: In this case, both $\mathcal{P}(\mathbf{x})$ and $\mathcal{P}(y|\mathbf{x})$ remain the same. By definition, the expected error will not change. However, it is still useful to train a model on the most recent data since the old model may not have achieved the minimum yet (such that there are examples with $y_p \neq y_M$) due to many reasons, such as the original training data is not sufficient that incurs variance in the model's prediction.

Feature Change: Feature change happens when $\mathcal{P}(\mathbf{x})$ changes but $\mathcal{P}(y|\mathbf{x})$ remains the same. In other words, some previously infrequent feature vectors become more frequent, and vice versa. As $\mathcal{P}(\mathbf{x})$ changes, the expected error may move up, move down, or stay the same, and it depends on specific combination between $\mathcal{P}(\mathbf{x})$ and the corresponding $\mathcal{P}(y|\mathbf{x})$ values. Model reconstruction can improve the estimated probability on those examples whose $y_p \neq y_M$ in the past due to small number of training examples.

Conditional Change: $\mathcal{P}(\mathbf{x})$ remains the same but $\mathcal{P}(y|\mathbf{x})$ changes. Under this condition, however, the minimum expected error rate integrated over all instances could go either way. The reason is that the error on an individual feature vector is dependent on $\mathcal{P}(y|\mathbf{x})$, and could increase, decrease, or remain the same. Thus, expected error cannot be used as an indicator of conditional change. When $\mathcal{P}(y|\mathbf{x})$ evolves, it is normally necessary to reconstruct the model.

Consider the problem on how to rebuild the model to match the new conditional probability. If the size

of new data is large enough to train an accurate new model by itself, old examples are not expected to help due to the concept evolution. However, if the new data is trivial in size, the error of a model trained from new data mainly comes from the variance and more training data can help reduce the variance. Existing approaches solve this problem through either weighted combination of old examples or selection of consistent old examples [7].

Dual Change: Both $\mathcal{P}(\mathbf{x})$ and $\mathcal{P}(y|\mathbf{x})$ change. Depending on the combination of $\mathcal{P}(\mathbf{x})$ and $\mathcal{P}(y|\mathbf{x})$, the expected error could increase, decrease or remain unchanged, thus cannot be used to indicate dual change. Similar to conditional change, since $\mathcal{P}(y|\mathbf{x})$ has evolved, it is normally necessary to train a new model.

Summary There are two main observations from the above hypothetical analysis. First, there isn't a general correlation between expected error of the previous model and any types of concept-drifts of the data stream. Thus, observing changes in expected error is not a reliable indicator of concept-drift. On the other hand, even when the expected error of the previous model does not change, there is usually still room of improvement by proper model reconstruction. Thus, an effective stream-mining algorithm shall not assume any particular type of concept-drift, but ought to assimilate new data as soon as possible, and consistently produce highly accurate models. However, stream data with skewed distribution is hard to handle since there may not be enough examples for the minority class in new data. We discuss about this problem in section 3.

2.2 Descriptive Model vs. Generative Model

The stream classification methods generally fall into two categories: generative and descriptive. Generative methods, such as Naive Bayes and logistic regression, assume that $\mathcal{P}(y|\mathbf{x})$ follows certain form of distributions whose parameters are to be estimated from training set. On the other hand, descriptive methods, such as decision tree, make little or no assumptions about the true form of $\mathcal{P}(y|\mathbf{x})$. Instead, both the structure and parameters of the hypothesis are learnt from training set. When applying on real data, descriptive models are expected to be more accurate than generative models.

First, for many stream applications, we have little or no prior knowledge about the data distribution, so it is hard to predefine the true form of $\mathcal{P}(y|\mathbf{x})$. Also, the form of distribution may evolve as data continuously arrives. For example, the underlying distribution may change from Gaussian to Beta. In reality, we never know when and how the underlying form changes. Therefore, it is rather risky to assume a certain kind of distribution for the data. Usually, a wrong assumption about the distribution form can lead to poor performances of generative models.

Second, the training data may not be balanced. In some real applications, the class distribution is highly skewed, there are insufficient examples for minority class. Typically non-parametric models require larger amount of data than parametric models. However, it is hard to learn parameters accurately from limited examples. If the training examples are far from sufficient, the parametric model would overfit the data and have low generalization accuracy [9].

Therefore, descriptive methods provide a more favorable solution to stream classification problems. Building a descriptive model does not require prior knowledge about the form of data distribution. The data speaks for itself. Also, no matter how the underlying distribution changes, the descriptive model would be easily adapted to the new distribution.

2.3 Probability Estimation vs. Label Prediction

There are two ways to classify test examples. One is to directly classify them into several categories, i.e., predict on their class labels. The other approach is to estimate $\mathcal{P}(y|\mathbf{x})$ and choose the best threshold to optimize on some given criteria. We would favor the second approach for the following reasons.

First, it is typically not known in advance if a problem is deterministic or stochastic without domain knowledge, and a previously deterministic problem may evolve into a stochastic one. The labels we observe for stochastic problems are not deterministic, instead, follow a certain distribution $\mathcal{P}(y|\mathbf{x})$. A specific feature vector $\mathbf{x}$ may appear in the training data with class label c, but this does not imply that $\mathbf{x}$ always belongs to class c. An example could be assigned to several classes with different probabilities. For example, if the chance of precipitation is 30%, then there is 70% probability that no rain falls. In this case, estimation of $\mathcal{P}(y|\mathbf{x})$ is more reasonable than predicting class labels.

Second, the predicted labels may not be accurate. Categorizing one example into a wrong class will invoke a huge amount of loss in some cases. Probability estimates, on the other hand, provide some information about uncertainties in classification. To ensure the accuracy of probability estimates, some calibration and smoothing methods could be used [16]. We would have high confidence in the prediction of an example with 99% posterior probability while are not sure about the prediction on an example with estimated posterior probability around 50%. This uncertainty information is very useful in decision making.

Finally, estimation of posterior probability provides a more flexible analysis scheme. Subjective or objective

criteria could be applied on probability estimates to find out the optimal decision threshold. For example, the threshold could be chosen so that the resulting precision and recall are balanced.

In summary, we argue that in real-world stream classification problems, a descriptive model that can output accurate probability estimates is preferred. The main reason is that it covers many situations of data streams than any methods we are aware of.

2.4 Desiderata In the above, we have argued that (1) concept-drift could happen in each sub-component of the joint probability distribution of the data stream, (2) expected error is not a reliable indicator of concept-drift, (3) mining new data chunk is expected to improve accuracy in most situations, (4) descriptive model is preferred over generative model for general purpose stream mining over a wide variety of problems, and (5) estimating probability is more suitable without strong assumption about the unknown true probability distribution. Although these observations are applicable to many situations of predictive mining in data streams, for the rest of this paper, we extend these ideas to design and evaluate a general framework to mine skewed concept-drifting data streams.

3 Mining Skewed Data Stream

In this section, we propose a simple strategy that can effectively mine data streams with skewed distribution. The choice of methods incorporates the analysis made in Section 2. Also, we provide a formal analysis on several aspects of the proposed framework.

3.1 Stream Ensemble Framework Skewed distribution can be seen in many data stream applications. In these cases, the positive examples are much less popular than the negative ones. Also, misclassifying a positive example usually invokes a much higher loss compared to that of misclassifying a negative example. Therefore, the traditional inductive learner, which tends to ignore positive examples and predict every example as negative, is undesirable for skewed stream mining. To handle skewed class distribution, we propose a simple, systematic method that applies on both deterministic and stochastic data streams. We will start with problem definition, and then present the algorithm.

In some applications such as credit card application flow, the incoming data stream arrives in sequential chunks, $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_m$ of the same size n. $\mathbf{S}_m$ is the most up-to-date chunk. The data chunk that arrives next is $\mathbf{S}_{m+1}$, and for simplicity, we denote it as $\mathbf{T}$. The aim of stream classification is to train a classifier based on the data arrived so far to estimate posterior probabilities of examples in $\mathbf{T}$. We further assume that the data comes from two classes, positive and negative classes, and the number of examples in negative class is much greater than the number of positive examples. In other words, $\mathcal{P}(+) \ll \mathcal{P}(-)$. In this two-class problem, only the posterior probability of positive class $\mathcal{P}(+|\mathbf{x})$ is computed, then that of the negative class is simply $1 - \mathcal{P}(+|\mathbf{x})$. To have accurate probability estimation, we propose to utilize both sampling and ensemble techniques in our framework.

Sampling. We split each chunk $\mathbf{S}$ into two parts $\mathbf{P}$, which contains positive examples in $\mathbf{S}$, and $\mathbf{Q}$, which contains negative examples in $\mathbf{S}$. The size of $\mathbf{P}$ is much smaller than that of $\mathbf{Q}$. For example, in network intrusion detection data, there are 60262 normal examples, but only 168 U2R attacks. Also, it should be noted that in stochastic problems, a given $\mathbf{x}$ could appear in both $\mathbf{P}$ and $\mathbf{Q}$ for several times. The count of $\mathbf{x}$ in each class will contribute to the calculation of posterior probability $\mathcal{P}(y|\mathbf{x})$.

In stream mining, we cannot use all data chunks as training data. First, stream data is huge in amount and it is usually impossible to store all of them. Second, stream mining requires fast processing, but a huge training set will make the classification process extremely slow, thus is unsatisfactory. In Section 2, we show that model reconstruction on new data reduces the expected error. In other words, the best way to construct a model is to build it upon the most recent data chunk. This works for examples in negative class since these examples dominate the data chunk and are sufficient for training an accurate model. However, the positive examples are far from sufficient. An inductive learner built on one chunk will perform poorly on positive class. To enhance the set of positive examples, we propose to collect all positive examples and keep them in the training set. Specifically, the positive examples in the training set are $\{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_m\}$. On the other hand, we randomly under sample the negative examples in the last data chunk $\mathbf{Q}_m$ to make the class distribution balanced. Though the strategy is quite simple, it is effective in skewed classification, as shown in Section 3.2.

Ensemble. Instead of training a single model on this training set, we propose to generate multiple samples from the training set and compute multiple models from these samples. The advantage of ensemble is that the accuracy of multiple model is usually higher than that of a single model trained from the entire dataset. The error of an inductive learner comes from both bias and variance. As shown in the next section, the variance could be reduced by training multiple models. The samples should be as uncorrelated as possible so that the base classifiers would make uncorrelated errors which

Ensemble algorithm:
Input: Current data chunk $\mathbf{S}$, test data $\mathbf{T}$, number of ensembles k, distribution ratio r, set of positive examples $\mathbf{AP}$
Output: Updated set of positive examples $\mathbf{AP}$, posterior probability estimates for examples in $\mathbf{T}$, $\{f^E(\mathbf{x})\}_{\mathbf{x}\in\mathbf{T}}$.
Algorithm:

1. Split $\mathbf{S}$ into $\mathbf{P}$ and $\mathbf{Q}$ according to their definitions.
2. Update $\mathbf{AP}$ as $\{\mathbf{AP}, \mathbf{P}\}$
3. Calculate the number of negative examples in the sample n_q based on the values of r and n_p.
4. **for** $i = 1$ to k **do**
   (a) Draw a sample of size n_q from $\mathbf{Q}$ without replacement, $\mathbf{O}$.
   (b) Train a classifier C_i on $\{\mathbf{O}, \mathbf{AP}\}$.
   (c) Compute posterior probability estimates $\{f^i(\mathbf{x})\}_{\mathbf{x}\in\mathbf{T}}$ using C_i
5. Compute posterior probability estimates by combining ensemble outputs $\{f^E(\mathbf{x})\}_{\mathbf{x}\in\mathbf{T}}$ based on Eq. (3.1).

Figure 1: Ensemble algorithm framework

could be eliminated by averaging. To get uncorrelated samples, each negative example in the training set is randomly propagated to exactly one sample, hence the negative examples in the samples are completely disjoint. As for positive examples, they are propagated to each sample. We take a parameter r as input, which is the ratio of positive examples over negative examples in each sample. r is typically between 0.3 to 0.6 to make the distribution balanced. Let n_p be the number of positive examples in the training set, then the number of negative examples in each sample is: $n_q = \lceil n_p/r \rceil$. Suppose k samples are generated, then a series of classifiers $C_1, C_2, \ldots, C_k$ are trained on the samples. Each classifier C_i outputs an estimated posterior probability $f^i(\mathbf{x})$ for each example $\mathbf{x}$ in $\mathbf{T}$. We use simple averaging to combine probability outputs from k models:

$$f^E(\mathbf{x}) = \frac{1}{k}\sum_{i=1}^{k} f^i(\mathbf{x}) \tag{3.1}$$

It is worth noting that this differs from bagging: 1) in bagging, bootstrap samples are used as training sets, and 2) bagging uses simple voting while our framework generates averaged probability for each test example. The outline of the algorithm is given in Figure 1. We assume that each data chunk can fit the main memory.

3.2 Analysis of Ensemble Framework We explain how the use of sampling and ensemble techniques contributes to error reduction. Also, we analyze the complexity of the algorithm.

3.2.1 Error Decomposition We expect that a well trained classifier could approximate the posterior class distribution. However, the estimate of posterior probability is not necessarily the true probability. Therefore, in classification, besides Bayes error, there are remaining errors, which could be decomposed into bias and variance. The bias measures the difference between the expected probability and the true probability, whereas the variance measures the changes in estimated probabilities using varied training sets. As stated in [12, 14], given $\mathbf{x}$, the output of a classifier can be expressed as:

$$f_c(\mathbf{x}) = \mathcal{P}(c|\mathbf{x}) + \beta_c + \eta_c(\mathbf{x}) \tag{3.2}$$

where $\mathcal{P}(c|\mathbf{x})$ is the posterior probability of class c given input $\mathbf{x}$, β_c is the bias introduced by the classifier and $\eta_c(\mathbf{x})$ is the variance of the classifier given input $\mathbf{x}$.

In two-class problem, $\mathbf{x}$ is assigned to positive class if $\mathcal{P}(+|\mathbf{x}) > \mathcal{P}(-|\mathbf{x})$. The Bayes optimal boundary is therefore represented by a set of points $\mathbf{x}^*$ that satisfy $\mathcal{P}(+|\mathbf{x}^*) = \mathcal{P}(-|\mathbf{x}^*)$. However, since $f_c(\mathbf{x})$ is different from $\mathcal{P}(c|\mathbf{x})$, the estimate of Bayes boundary is incorrect, the boundary error is $b = \mathbf{x}_b - \mathbf{x}^*$ where $\mathbf{x}_b$ are the estimated boundary points that have $f_+(\mathbf{x}_b) = f_-(\mathbf{x}_b)$. In [12], it shows that classification error rate is linearly proportional to the boundary error. So we will focus on the analysis of boundary error from now on. In analogy with bias-variance decomposition described in Eq. (3.2), the boundary error can be expressed in terms of boundary bias and boundary variance:

$$b = \frac{\eta_+(\mathbf{x}_b) - \eta_-(\mathbf{x}_b)}{s} + \beta_b \tag{3.3}$$

where $s = p'_+(\mathbf{x}^*) - p'_-(\mathbf{x}^*)$ is independent of the trained model, and β_b is $(\beta_+ - \beta_-)/s$. If $\eta_c(\mathbf{x})$ is independent and has Gaussian distribution with zero mean and variance $\sigma^2_{\eta_c}$, then b is also normally distributed with mean β_b and variance σ_b^2 where

$$\sigma_b^2 = (\sigma^2_{\eta_+} + \sigma^2_{\eta_-})/s^2 \tag{3.4}$$

3.2.2 Error Reduction by Sampling We show that sampling techniques in the proposed framework reduces variance in skewed data classification.

Our sampling approach could reduce σ_b^2 not at the expense of increase in β_b. If only the current data chunk is used to train the model, the positive examples are so limited that the error of the classifier would mainly come from the variance. In the proposed framework, the positive examples in the previous time shots are incorporated into the training set. Adding positive examples would reduce the high variance σ_b^2 caused by insufficient data. When there are concept changes, the bias may be affected by adding old examples, but it may

increase very slightly. The reason is that the negative examples of the training set are from the current data chunk, which are assumed sufficient and reflecting the current concept. Therefore, the boundary between the two classes could not be biased much by including old positive examples in the training set. Even if the bias β_b is increasing, the reduction of variance is dominant and the overall generalization accuracy is improved.

3.2.3 Error Reduction by Ensemble The use of ensemble could further reduce the variance of single classifiers. According to Eqs. (3.1) and (3.2), the following formula holds on:

$$f_c^E(\mathbf{x}) = \mathcal{P}(c|\mathbf{x}) + \overline{\beta}_c + \overline{\eta}_c(\mathbf{x}) \tag{3.5}$$

where $\overline{\beta}_c$ and $\overline{\eta}_c(\mathbf{x})$ are average bias and variance respectively. If the noise error of each ensemble is independent, the variance of $\overline{\eta}_c(\mathbf{x})$ is:

$$\sigma^2_{\overline{\eta}_c} = \frac{1}{k^2}\sum_{i=1}^{k}\sigma^2_{\eta_c^i} \tag{3.6}$$

Based on Eq. (3.4), it can be derived that the boundary variance can be reduced by a factor of k^2:

$$\sigma^2_{b^E} = \frac{1}{k^2}\sum \sigma^2_{b^i}$$

Therefore, our proposed framework would greatly reduce variance by employing both sampling and ensemble techniques into skewed stream classification. As shown in the experiments, the improvements gained in accuracy are significant.

3.2.4 Efficiency Analysis In addition to error reduction, the benefits of ensemble over single classification model involve efficiency improvements. Suppose the base learner is decision tree and the dimension of data is d. As denoted above, the number of positive examples in each sample is n_p and the number of negative examples is n_q. Then the training size of each sample is $n_p + n_q$, so the time complexity of a decision tree-based learner would be $O(d(n_p+n_q)\log(n_p+n_q))$. If these ensembles are executed sequentially, the training time for multiple model would be $O(dk(n_p+n_q)\log(n_p+n_q))$. If we train a single model on the whole data set, the training size is n_p+kn_q, then the time complexity is $O(d(n_p+kn_q)\log(n_p+kn_q))$. Since the class distribution in each sample is balanced, we assume that $n_p = n_q = n_e$. Then the time complexity of single model and ensemble could be represented by $O(d(k+1)n_e\log((k+1)n_e))$ and $O(2dkn_e\log(2n_e))$ respectively. k is the number of models in the ensemble, and is typically greater than 3. Since $\log(k+1) > 2\log 2$ and $k+1 > k$, we can conclude that the ensemble is more efficient than the single model even if the ensemble is executed in sequence. In reality, since each classifier in the ensemble is independent, they can be computed in parallel. The gain in efficiency would be more significant for parallel ensembles.

4 Experiments

We have conducted thorough experiments on both synthetic and real data sets. We analyze the proposed stream ensemble method from the following perspectives: (1) our proposed method would have good performances no matter how the concept changes, $\mathcal{P}(\mathbf{x})$, $\mathcal{P}(y|\mathbf{x})$ or both; (2) for a series of real data sets where the forms of distributions and concept changes are both unknown, the stream ensemble method we propose is able to provide accurate probability estimates; (3) with reliable estimation of posterior probability, our method could gain great improvements on prediction accuracy; (4) besides gains in accuracy, the efficiency is also improved through ensemble framework.

4.1 Experiment Setup

4.1.1 Synthetic Data Generation We generate synthetic data streams with different kinds of concept changes. The data is of the form $(\mathbf{x}, y)$ where $\mathbf{x}$ is a multi-dimensional feature vector and $y \in \{0, 1\}$ is the label of the example. We describe in the following how we simulate $\mathcal{P}(\mathbf{x})$, $\mathcal{P}(y|\mathbf{x})$, and their changes.

Form of $\mathcal{P}(\mathbf{x})$. $\mathbf{x}$ follows a Gaussian distribution, i.e., $\mathcal{P}(\mathbf{x}) \sim N(\mu, \mathbf{\Sigma})$, where μ is the mean vector and Σ is the covariance matrix. The feature change is simulated through the change of the mean vector. Suppose μ_i is the mean on the i-th dimension, then it is changed to $\mu_i s_i(1+t)$ for each data chunk. Specifically, t is between 0 to 1, representing the magnitude of changes, $s_i \in \{-1, 1\}$ specifies the direction of changes and could be reversed with a probability of 10%.

Form of $\mathcal{P}(y|\mathbf{x})$ in deterministic problems. The probability of having a label y is either 1 or 0 in this case. Let x_i be the value of $\mathbf{x}$ on the i-th dimension, and a_i be the weight assigned to the corresponding dimension. In [14], a hyperplane is used to characterize the boundary between two classes, which is defined as a function of x_i and a_i. In this experiment, a more complicated boundary is used to generate datasets which are difficult to learn. The boundary is defined using function $g(\mathbf{x}) = \sum_{i=1}^{d} a_i x_i x_{d-i+1} - a_0$. Then the examples satisfying $g(\mathbf{x}) < 0$ are labeled positive, whereas other examples are labeled negative. Weights $a_i (1 \le i \le d)$ are initialized by random values in the range of [0,1]. We set the value of a_0 so that the number of positive examples is much smaller than that

Table 1: Description of Data Sets

data sets	two classes	#inst	#feature	#rare class inst	#chunk	chunksize
Thyroid1	Class 1 vs. Class 3	6832	21	166	6	1138
Thyroid2	Class 2 vs. Class 3	7034	21	368	6	1172
Opt	each class vs. rest	5620	64	554-572	6	936
Letter	each class vs. rest	20000	16	131	6	3332
Covtype	Class 2 vs. Class 4	28604	54	274	11	2599

of negative examples. A skewness ratio r is used to control the degree of skewness. The concept change in $\mathcal{P}(y|\mathbf{x})$ is represented by the change in weight a_i. a_i is changed to $a_i s_i(1+t)$ for every data chunk, where the parameters t and s_i are defined in the same way as in the feature change described above.

Form of $\mathcal{P}(y|\mathbf{x})$ in stochastic problems. The label could be stochastically generated, where $\mathcal{P}(y|\mathbf{x})$ is between 0 and 1 for each y. We use a sigmoid function to model the posterior distribution of positive class:

$$\mathcal{P}(+|\mathbf{x}) = \frac{1}{1+\exp(g(\mathbf{x}))}$$

The skewness is also controlled by r as in deterministic problems. In fact when $g(\mathbf{x}) = 0$, the posterior probability of $\mathbf{x}$ belonging to positive class is 0.5. Therefore, if r is small, e.g., 0.01, examples that have 0.5 or above posterior probability only account for 1%. The concept changes are also realized by the changes of weights as illustrated in the deterministic scenario.

We generate b data chunks of size n. As stated in Section 2, there are three kinds of changes: feature change, conditional change, and dual change. The changes are simulated by adjusting the parameters as described above. The distribution within a data chunk is unchanged. Between data chunks, either one of the three changes occurs.

4.1.2 Real Data Sets Besides synthetic data, we use a series of real data sets from UCI machine learning repository. Although these data sets do not directly correspond to skewed data mining problems, they can be converted into rare class problems by taking one small class as the rare class and taking the remaining records or the biggest remaining class as the second class. To simulate a data stream, the data is randomly partitioned into several chunks with skewed distribution maintained in each chunk. The data sets are summarized in Table 1.

4.1.3 Measures We would evaluate the proposed method in two perspectives: (1) are the probability estimates accurate? and (2) is the classification accurate?

There are some standard measures for evaluating the quality of probability estimation. A popular one is mean squared error, defined as:

$$L = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - \mathcal{P}(+|\mathbf{x}_i))^2 \tag{4.7}$$

where $f(\mathbf{x}_i)$ is the output of ensemble, which is the estimated posterior probability of $\mathbf{x}_i$, and $\mathcal{P}(+|\mathbf{x}_i)$ is the true posterior probability of $\mathbf{x}_i$. Since in skewed mining problems, rare class is more interesting and usually associated with higher classification cost, we would like to have a low L for examples in the rare class.

In skewed mining problems, classification error is not a good measure since the examples in the majority class will dominate the result, and it is hard to tell whether rare examples are classified correctly. Therefore, for this kind of problems, the following evaluation metrics are typically used: Precision (Detection Rate), Recall, and False Alarm Rate. To show how these metrics are correlated, we use both ROC curve and recall-precision plot to demonstrate the experimental results. The ROC curve represents the trade-off between detection rate and false alarm rate and plots a 2-D graph, with x-axis as the false alarm rate and y-axis as the detection rate. The ideal ROC curve has 0% false alarm rate and 100% detection rate. In other words, the area under ROC curve is 1 in the ideal case. Therefore, a good algorithm would produce a ROC curve as close to the left-top corner as possible. So the area under ROC curve (AUC) is an evaluation metric, where a better algorithm will have an AUC value closer to 1. Another method to evaluate the results is to plot the correlation between recall and precision. The recall-precision plot will have precision as the y axis and recall as the x axis.

4.1.4 Baseline Methods In Section 3.2, we show that both sampling and ensemble techniques could help reduce the classification error. Therefore, the baseline methods we are comparing with are:

No Sampling + Single Model (*NS*). Only the current data chunk is used for training, which is highly skewed. A single model is trained on the training set.

Sampling + Single Model (*SS*). The training set is the same as that used in our proposed ensemble

Table 2: Mean Squared Error on Deterministic Stream Data

Changes	Decision Trees			Naive Bayes			Logistic Regression		
	SE	NS	SS	SE	NS	SS	SE	NS	SS
Feature	0.1275	0.9637	0.6446	**0.0577**	0.8693	0.4328	0.1501	0.8117	0.5411
Conditional	0.0943	0.9805	0.5500	**0.0476**	0.8830	0.4380	0.1301	0.8944	0.5729
Dual	0.0854	0.9521	0.5174	**0.0664**	0.8596	0.4650	0.1413	0.8371	0.5525

Table 3: Mean Squared Error on Stochastic Stream Data

Changes	Decision Trees			Naive Bayes			Logistic Regression		
	SE	NS	SS	SE	NS	SS	SE	NS	SS
Feature	0.0847	0.6823	0.4639	**0.0314**	0.5371	0.2236	0.0974	0.5311	0.3217
Conditional	0.0552	0.6421	0.4463	**0.0299**	0.5675	0.2449	0.1029	0.6578	0.4151
Dual	0.0684	0.6758	0.4107	**0.0301**	0.5981	0.2556	0.0887	0.6388	0.4075

methods. It is obtained by keeping all positive examples seen so far and under sampling negative examples in the current data chunk. The difference lies in the classification model which is a single model in this method, but a multiple model in our proposed method.

Accordingly, we denote our method as *Sampling + Ensemble* (*SE*), which adopts both sampling and ensemble techniques. By comparing with the above two baseline methods, the strengths of sampling and ensemble could be illustrated well.

In the experiments, the base learners include both parametric and non-parametric classifiers: Decision Tree, Naive Bayes and Logistic Regression. We use the implementation in Weka package [13]. The parameters for single and ensemble models are set to be the same, which are the default values in Weka.

4.2 Empirical Results In this part, we report the experimental results regarding the effectiveness and efficiency of our proposed method. The results demonstrate that the ensemble method could improve both the quality of probability estimates of the positive class and the classification accuracy in terms of the ROC curve and recall-precision plot, and is efficient according to training time comparison.

4.2.1 Test on Concept-Drift Streams We generate synthetic data streams with different kinds of concept drifts. Six kinds of stream data sets are generated, each of which is either deterministic or stochastic, and has either feature, conditional, or dual concept changes. Each data set has 10 dimensions, 11 chunks with chunk size 1000. The percentage of rare examples is 1%, and for each data chunk, two dimensions are chosen randomly to change 10%. The last chunk in the data set is recognized as the test data, and all other data chunks are used for training. Since we are more interested in probability estimates of the positive class, the mean square error of the positive class is reported for each kind of stream data. The results are obtained by calculating errors of 10 randomly generated data sets. We use C4.5, Naive Bayes, and logistic regression as base learners. The results are shown in Table 2 (deterministic) and Table 3 (stochastic), respectively.

It is clearly seen that, no matter how the concept changes, our proposed method (SE) greatly improves the mean square error of the positive class in both deterministic and stochastic data streams. The decrease in error rate is significant, from 0.9 to 0.1 on the deterministic data, and from 0.6 to 0.06 on the stochastic data on average. NS performs badly since only the current data chunk is used for training and it is highly skewed. When training on a skewed data set, the inductive learner would build a model that tends to ignore positive examples and simply classify every example as negative. Therefore, NS generates an error close to 1 regarding mean square error on the positive class. According to the performances of SS, the mean square error of the positive class is reduced to around 0.5 after we oversample positive examples and under sample negative examples. The reason is that the class distribution is more balanced after incorporation of more positive examples. This helps improve the performances on the positive class. However, the single model would still have a high error rate caused by classification variance.

Although SE utilizes exactly the same training sets as used in SS, the performances of SE are much better since the ensemble could reduce the variance of classifier by averaging the outputs. As seen in Tables 2 and 3, for our proposed method, the mean square error on the positive class is usually less than 0.1. The most significant reduction in error rate is 0.45 (SE vs. SS) and 0.88 (SE vs. NS). On average, the error decreases around 40% after using sampling and further reduces to 10%-20% if we use both sampling and ensemble.

It can be observed that Naive Bayes has the best

Table 4: Mean Squared Error on Real Data

Data Set	Decision Trees			Naive Bayes			Logistic Regression		
	SE	NS	SS	SE	NS	SS	SE	NS	SS
Thyroid1	**0.0000**	0.0799	0.0003	0.0057	0.0655	0.0366	0.0105	0.2001	0.0634
Thyroid2	**0.0001**	0.0266	0.0047	0.0505	0.4774	0.2323	0.0044	0.2443	0.0391
Opt	0.0147	0.1160	0.0414	0.0106	0.0972	0.0106	**0.0025**	0.1131	0.0225
Letter	**0.0191**	0.2290	0.0653	0.1024	0.2459	0.1567	0.0595	0.4091	0.2061
Covtype	0.0003	0.2500	0.0834	**0.0000**	0.4496e-7	0.0001e-7	0.0008	0.0835	0.0417

performance on synthetic data set. This is due to the fact that synthetic data is generated using a Gaussian distribution with a diagonal covariance matrix, which guarantees independence between features.

Thus we conclude that our proposed method consistently improves posterior probability estimates of minority class under feature, conditional, and dual concept drifts in both deterministic and stochastic applications.

4.2.2 Test on Real Data In this part, we conduct experiments on several real life data sets. For "thyroid" data set, we select either class 1 or class 2 as the rare class and class 3 as the majority class. Similarly, for "covtype" data set, the biggest class—class 2 is identified as the majority class while the smallest class—class 4 corresponds to the rare class. Data sets "letter" and "opt" are used for recognition of 26 letters and 10 digits respectively. For these two data sets, we simply choose each class to represent the rare class and collapse the remaining classes into one majority class. By this way, from the original data sets, 26 and 10 skewed data sets are generated and the results are averaged over these generated data sets.

From Table 4, it can be observed that our proposed method SE consistently outperforms NS and SS on real life data sets. The improvements are not that significant compared to those on the synthetic data sets. This is probably due to the learning complexity of data sets. The concepts of synthetic data sets are changed every data chunk, which makes them harder to learn. NS and SS are both unable to handle these fast-changing, highly skewed data sets. By balancing the training data and using multiple models, the improvements of SE on the synthetic data are more apparent.

In real data sets, the concept changes are not intense, so the error rates of NS and SS are typically around 0.2 and less than 0.1 respectively. This leaves a small space for SE to improve. Nevertheless, SE successfully achieves an error less than one tenth of the error generated by NS most of the time. When NS and SS have high error rates of around 0.4 and 0.2, SE could reduce the error to 0.05. Another interesting observation is that no base learner is globally optimal.

Table 5: Decision Tree as Base Learner

	SE	NS	SS
Synthetic1	**0.9464**	0.5175	0.6944
Synthetic2	**0.9337**	0.4840	0.6611
Thyroid1	**1.0000**	0.9999	0.9999
Thyroid2	**0.9998**	**0.9998**	0.9996
Opt	**0.9942**	0.9495	0.9777
Letter	**0.9931**	0.9467	0.9782
Covtype	**1.0000**	**1.0000**	0.9999

Table 6: Naive Bayes as Base Learner

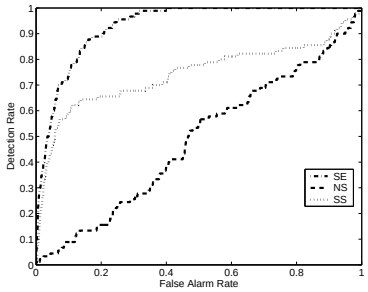
	SE	NS	SS
Synthetic1	**0.9532**	0.8220	0.9525
Synthetic2	**0.9558**	0.8355	0.9556
Thyroid1	**0.9982**	0.9979	**0.9982**
Thyroid2	**0.9551**	0.9054	0.9145
Opt	**0.9926**	0.9722	0.9898
Letter	**0.9395**	0.9389	0.9389
Covtype	**0.9997**	0.9995	**0.9997**

Table 7: Logistic Regression as Base Learner

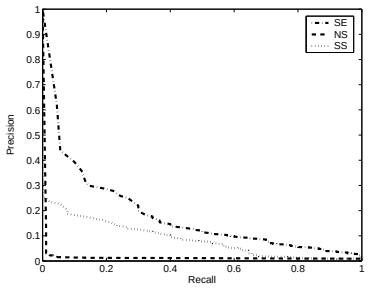
	SE	NS	SS
Synthetic1	**0.8801**	0.8363	0.8737
Synthetic2	**0.8992**	0.8102	0.8854
Thyroid1	**0.9977**	0.9774	0.9909
Thyroid2	**0.9949**	0.9593	0.9930
Opt	**0.9971**	0.9940	0.9953
Letter	**0.9545**	0.9448	0.9517
Covtype	**0.9995**	0.9989	0.9994

Decision trees have good performance on "thyroid" and "letter", Naive Bayes is suitable for "covtype", and "opt" is best classified by Logistic Regression. This demonstrates that these real data sets have different characteristics with different sizes, skewness degrees, and distributions. Such a diverse testbed shows the wide capabilities of our proposed method.

4.2.3 Model Accuracy Analysis The purpose of this experiment is to compare the model accuracy in terms of detection, recall, and false alarm rates. Tables 5 to 7 show the results of applying our proposed method and two baseline methods on a series of synthetic and

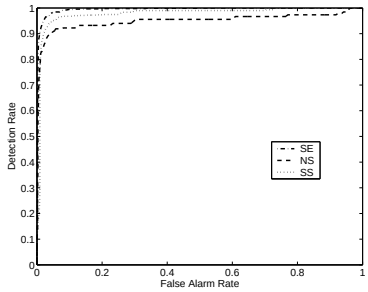


(a) ROC curve

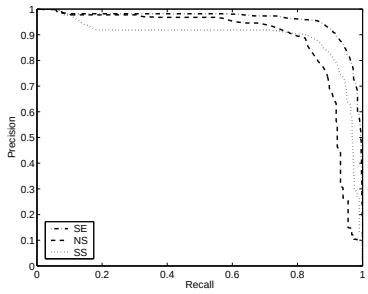


(b) Recall-Precision Plot

Figure 2: Plots on Synthetic1



(a) ROC curve



(b) Recall-Precision Plot

Figure 3: Plots on Opt

real data sets. The measure is the area under ROC curve (AUC). The data sets "synthetic1" and "synthetic2" refer to deterministic and stochastic data sets with dual concept change generated as described above. Again, the greatest enhancements are achieved on synthetic data sets. When the base learner is Decision Tree, SE increases AUC to over 0.9 while AUC of NS and SS are only around 0.5 and 0.6. On real data sets, NS has already gained an AUC over 0.9. Yet, our proposed method is still consistently better than its competitors.

We further show the ROC curves and recall-precision plots of data sets synthetic1 and Opt in Figures 2 and 3. The parameter setting is the same as described above and the base learner is C4.5. Clearly, in both synthetic and real applications, our proposed method consistently improves precision. The synthetic data set is hard to learn due to highly skewed distribution and concept changes, thus the precision obtained by single model is extremely low, around 0.02 in the recall range of [0.4 0.6]. Our method has achieved nearly ten times as much precision as the single model and an ROC curve with an area near 1. It justifies that the sampling and ensemble techniques used in our method effectively improve the probability estimates and thus lead to a better classification model. The improvements in Opt data set are not that significant. The reason is that the data set is not sharply skewed (nearly 10% of positive examples), so the single model has already obtained a high precision (nearly 90%). However, our proposed method succeeds in improving both ROC and recall-precision graphs by 10% increase in precision since the classification variance is reduced.

4.2.4 Effect of Data Size and Skewness First, we study the impact of chunk size on the reliability of probability estimates. The data sets we used are synthetic1 with skewness degree 0.01 and the base learner is C4.5. We vary the chunk size from 1000 to 5000. The results are shown in Table 8. In general, the classification errors of all three methods decrease with respect to both MSE and AUC when chunk size increases because more training examples lead to reduction in classification errors. But in real applications, the chunk size is usually fixed, which is the number of examples that could be held in memory. It is noted that SE always outperforms NS and SS when chunk size varies. For example, when chunk size increases from 1000 to 5000, MSE of SE decreases from

Table 8: Effect of Chunk Size

Chunk-size	MSE			AUC		
	SE	NS	SS	SE	NS	SS
1000	**0.0731**	0.6999	0.3575	**0.9259**	0.4991	0.7384
2000	**0.0533**	0.6759	0.3680	**0.9522**	0.5379	0.7555
3000	**0.0556**	0.7015	0.3354	**0.9557**	0.5061	0.7578
4000	**0.0478**	0.6749	0.3565	**0.9643**	0.5580	0.7432
5000	**0.0423**	0.6243	0.3107	**0.9710**	0.5913	0.7425

Table 9: Effect of Skewness

Skew-ness	MSE			AUC		
	SE	NS	SS	SE	NS	SS
0.01	**0.0401**	0.6862	0.3486	**0.9527**	0.4820	0.7417
0.02	**0.0461**	0.6852	0.3517	**0.9623**	0.5107	0.7433
0.03	**0.0424**	0.6481	0.3235	**0.9688**	0.6175	0.7422
0.04	**0.0383**	0.6071	0.3238	**0.9718**	0.6281	0.7535
0.05	**0.0346**	0.6190	0.2810	**0.9750**	0.5954	0.7773

0.0731 to 0.0423 while MSE of NS is much higher, changing from 0.6999 to 0.6243.

Then, we show the effects of skewness on classification error. The data set is synthetic1 with chunk size 1000 and base leaner is decision tree. According to Table 9, classification errors of NS decrease when the percentage of positive examples increases from 0.01 to 0.05. For example, AUC of SS increases from 0.4820 to 0.5954. The reason is that for NS, though the training size does not change, more positive examples would help reduce the classification error. The same trend is observed for both SS and SE, whose AUC increases from 0.7417 to 0.7773 and from 0.9527 to 0.9750, respectively. Although training sets in SS and SE are made balanced by sampling, the skewness degree in the original data set would still have impacts on the training size. If the percentage of positive examples is low, we need to greatly under sample the negative examples to make distribution balanced. However, if the percentage of positive examples is high, we could incorporate more negative examples into the training set. With a larger training set, a more accurate classification model is expected.

4.2.5 Training Efficiency We study the time complexity of the ensemble approach compared with single model. A synthetic data stream with 10 chunks is generated. The chunk size is varied from 1000 to 5000 and the number of ensembles is 5. The training times of both parallel and serial ensembles are reported in Figure 4. As expected, ensemble outperforms the corresponding single model significantly, especially when the chunk size is large. With chunk size 1000, serial ensemble reduces the training time by half. The reduction becomes more significant as chunk size increases. With chunk size 5000, the training time of single model is four times

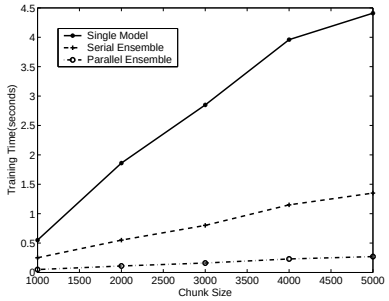


Figure 4: Training Time

the training time of serial ensemble. When the training set has a high cardinality, training a single model on this huge data set would cost a large amount of time. The ensemble method, on the other hand, divides the large training set into several small sets, and thus saves lots of time on training even though these classifiers are trained sequentially. The gain in efficiency is more significant if the classification processes are executed in parallel. The results verify that our proposed method not only improves on classification accuracy, but also on efficiency and scalability.

5 Related Work

Class imbalance has become an important research problem in recent years since more people have realized that imbalance in class distribution causes suboptimal classification performance [15, 5]. Several workshops, such as those at AAAI 2000 and ICML 2003, and a journal special issue at SIGKDD Exploration 2004 are dedicated to class imbalance problem. Many solutions have been proposed to handle this problem by preprocessing data, transforming algorithms or post-processing models [1]. Among them, balancing training set distribution is the most popular approach. Specifically, many sampling algorithms have been developed to either under sample majority examples or oversample minority examples [4, 1]. These methods may improve the prediction accuracy of minority class, however, they are greatly challenged by stream applications where infinite data flow and continuous concept drifts are present. Therefore, a general framework for dealing with skewed data stream is in great demand.

Skewed stream problems have been studied in the context of summarization and modeling [6, 11]. However, the evaluation of existing stream classification methods is done on balanced data streams [10, 14, 7, 2]. In reality, the concepts of data streams usually evolve with time. Several stream classification models are designed to mine such concept-drifting data streams [14, 7], however, they regard concept drifts as changes

in conditional probability. In our work, it is shown that concept changes may occur in both feature and conditional probability. In [8, 17], two application examples of skewed data mining are studied. But we provide a more general framework for building accurate classification models on skewed data streams.

6 Conclusions

This paper has two important and related parts. In the first part, we analyze several requirements and choices of techniques for general predictive modeling on data streams, which are not clearly understood previously. A comprehensive understanding of these requirements helps design better algorithms and evaluate the performance of these algorithms in an environment close to reality. To be specific, first, we argue that concept drifts of data streams could involve changes in either $\mathcal{P}(\mathbf{x})$ or $\mathcal{P}(y|\mathbf{x})$. We show that when concept evolves in either form, the expected error could decrease, increase, or stay the same. However, the most up-to-date data could always help reduce classification error. Second, we explain why descriptive models should be preferred over generative models for stream mining. Descriptive models could represent the true concept more accurately especially in data streams because: (1) many applications have skewed distribution, and (2) both the underlying true distributions and concept evolution patterns remain unknown either before or after mining. We also discuss about the benefits to use posterior probability estimation as compared to direct class label prediction for data stream classification. The probability estimates provide more refined information to users for better decision making. Also, for stochastic problems, estimation of $\mathcal{P}(y|\mathbf{x})$ is more meaningful.

Although these analyses are helpful for inductive learning on many stream mining problems, we are particularly interested in applying them to mine skewed data streams. We design an effective framework based on sampling and ensemble techniques. The algorithm first generates a balanced training set by keeping all positive examples and under sampling negative examples. Then the training set is further divided into several samples and multiple models are trained on these samples. The final outputs are the averaged probability estimates on test data by multiple models. We show that both sampling and ensemble techniques contribute to classification variance reduction. The error reduction is significant according to experimental results, e.g., for concept-drifting streams, the mean square error decreases from around 0.9 to 0.1. It is also demonstrated in both formal analysis and experiments on synthetic and real-world datasets that the proposed method is not only effective in error reduction, but also efficient and scalable with respect to training time.

The method introduced in this paper mainly focuses on two-class problems, but it could be easily extended to multi-class problems. Directions for future work include applying the framework to multi-class data with skewed class distributions and analyzing the strengths of ensemble methods in stream environments with various kinds of concept changes.

References

[1] N. Abe. Sampling approaches to learning from imbalanced datasets: active learning, cost sensitive learning and beyond. In *Proc. of ICML-KDD'03 Workshop: Learning from Imbalanced Data Sets*.

[2] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu. On demand classification of data streams. In *Proc. of KDD '04*.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of PODS '02*.

[4] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1), 2004.

[5] N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1), 2004.

[6] G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *Proc. of SDM'05*.

[7] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proc. of KDD '04*.

[8] W. Fan, P. S. Yu, and H. Wang. Mining extremely skewed trading anomalies. In *Proc. of EDBT'04*.

[9] T. Hastie, R. Tibshirani, J. Friedman. The Elements of Statistical Learning. *Springer-Verlag*, 2001.

[10] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of KDD '01*.

[11] F. Korn, S. Muthukrishnan, and Y. Wu. Modeling skew in data streams. In *Proc. of SIGMOD '06*.

[12] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2),1996.

[13] I. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques. *Morgan Kaufmann*, 2005.

[14] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of KDD '03*, 2003.

[15] G. Weiss and F. Provost. The effect of class distribution on classifier learning. Technical Report ML-TR-43, Rutgers University,, 2001.

[16] J. Zhang and Y. Yang. Probabilistic score estimation with piecewise logistic regression. In *Prof. of ICML '04*, 2004.

[17] K. Zhang, W. Fan, X. Yuan, I. Davidson, and X. Li. Forecasting skewed biased stochastic ozone days. In *Proc. of ICDM'06*.