An Adaptive Learning Approach for Noisy Data Streams

Fang Chu   Yizhou Wang   Carlo Zaniolo
Technical Report, TR040029
Computer Science Department, UCLA
{fchu,wangyz,zaniolo}@cs.ucla.edu

Abstract

Two critical challenges typically associated with mining data streams are concept drift and data contamination. To address these challenges, we seek learning techniques and models that are robust to noise and can adapt to changes in timely fashion. In this paper, we approach the stream-mining problem using a statistical estimation framework, and propose a fast and robust discriminative model for learning on noisy data streams. We build an ensemble of classifiers to achieve timely adaptation by weighting classifiers in a way that maximizes the likelihood of the data. We further employ robust statistical techniques to alleviate the problem of noise sensitivity. Experimental results on both synthetic and real-life data sets demonstrate the effectiveness of this new model learning approach.

1. Introduction

There is much current research interest in continuous mining of data: i.e., data that arrives continuously in high volume and speed. Applications involving stream data abound and include network traffic monitoring, credit card fraud detection and stock market trend analysis. Practical situations pose three fundamental issues to be addressed by any continuous mining attempt.

- **Adaptation Issue.**

  In traditional learning tasks, data is stationary and the underlying concept that maps the attributes to class labels is unchanging. With data streams, however, the concept is not static but drifts with time due to changes in the environment. For example, customer purchase preferences alter with seasons, fashion, and the emerging of competing products and services. These changes cause the model learned from old data obsolete and inconsistent with the new data, and updating the model becomes necessary. This is commonly known as the concept drift problem [20]. It is considered to be one of the core issues of data stream mining [8].

- **Robustness Issue.**

  Data contamination is a serious problem since noise can severely impair the quality and speed of learning. This problem is encountered in many applications where the source data can be unreliable, and also errors can be injected during data transmission. This problem is even more challenging for data streams, where it is difficult to distinguish noise from data caused by concept drift. If an algorithm is too eager to adapt to concept changes, it may overfit noise by mistakenly interpreting it as data from a new concept. If the algorithm is too conservative and slow to adapt, it may overlook important changes (and, for instance, miss out on the opportunities created by a timely identification of new trends in the marketplace).

- **Performance Issue.**

  Constrained by the requirement of on-line responses and by limited computation and memory resources, continuous data stream mining should conform to the following criteria: (1) learning should be done very fast, preferably in one pass of the data; and (2) algorithms should make light demands on memory resources, for the storage of either the intermediate results or the final decision models. These “fast and light” requirements exclude computationally expensive algorithms, such as support vector machines, or very large models, such as decision trees with thousands of nodes.

1.1. Related Work

The concept drift problem has been addressed in both machine learning and data mining communities. The first systems capable of handling concept drift were STAGGER [15], ib3 [2] and the FLORA family [20]. These algorithms provided valuable insights. But, as they were developed and tested only on small datasets, they may not be suitable for data streams which is on a significantly larger scale than previously studied.

Several scalable learning algorithms designed for data streams were proposed recently. They either maintain a single model incrementally, or an ensemble of base learners. The first category includes Hoeffding tree [10], which grows a decision tree node by splitting an attribute only when that attribute is statistically predictive. Hoeffding-tree like algorithms need a large training set in order to reach a fair performance, which makes them unsuitable to situations featuring frequent changes. Domeniconi and Gunopulos [7] designed an incremental support vector machine algorithm for continuous learning.

The second category of scalable learning algorithms are ensemble-based approaches. Kolter and Maloof [11] proposed to track concept drift by an ensemble of experts. The poor experts are weighted down or discarded, and the good experts are updated using recent examples incrementally. Their method is limited to situations where the values of each attribute follow a Gaussian distribution. This assumption is necessary to get an efficient incremental algorithm, but not practical in many situations.

Other ensemble-based approaches simply partition the data stream into sequential blocks of fixed size and learn an ensemble from these blocks. Base models are constructed once at a time and never updated incrementally, so that any off-the-shelf learning algorithm can be used. These ensembles adopt different voting schemes to reach a final decision. Street et al. [17] let their ensemble vote uniformly, while Wang et al. [18] prefer a weighted voting, assigning classifier weights proportional to their accuracy on the latest data block. These two approaches, [17] and [18], are most closely related to what we will present in this paper, as our method also builds an ensemble from sequential blocks. For ease of later reference in comparative study, we name them as *Bagging* and *Weighted Bagging*, respectively. (The name "*bagging*" derives from their analogy to traditional bagging ensembles [4].)

What is missing from the above mentioned work is a mechanism for noise identification, or often indistinguishably called *outlier* detection (as what we will use thereafter). Although there have been a number of off-line algorithms [1, 14, 5, 12, 6] for outlier detection, they are unsuitable for stream data as they assume a single unchanging data model, hence unable to distinguish noise from data caused by concept drift. In addition, outlier detection with stream data faces general problems such as the choice of a distance metric. Most of the traditional approaches use Euclidean distance, which is unable to treat categorical values.

1.2. Our Method

To address the three above mentioned issues, we propose a novel discriminative model for adaptive learning on noisy data streams, with modest resource consumption. For a learnable concept, the class of a sample conditionally follows a Bernoulli distribution. Our method assigns classifier weights in a way that maximizes the training data likelihood with the learned distribution. This weighting scheme has theoretical guarantee for adaptability. In addition, as we have verified experimentally, our weighting scheme can also boost a collection of weak classifiers into a strong ensemble. Examples of weak classifiers include decision trees with very few nodes. It is desirable to use weak classifiers because it learns faster and consumes less resources.

Our outlier detection differs from previous approaches in that it is tightly integrated into the adaptive model learning. The motivation is that outliers are directly defined by the current concept, so the outlier identifying strategy needs to be modified whenever the concept drifts away. In our integrated learning, outliers are defined as samples with a small likelihood given the current model, and then the model is refined on the training data with outliers removed. The overall learning is an iterative process in which the model learning and outlier detection mutually reinforce each other.

Another advantage of our outlier detection technique is the general distance metric for identifying outliers. We define a distance metric based on predictions of the current ensemble, instead of a function in the data space. It can handle both numerical and categorical values.

In section 2 and section 3 we describe the discriminative model with regard to adaptation and robustness, respectively. Section 4 gives the model formulation and computation. Experimental results are shown in section 5.

2. Adaptation to Concept Drift

Ensemble weighting is the key to fast adaptation. Here we show that this problem can be formulated as a statistical optimization problem solvable by logistic regression.

We first look at how an ensemble is constructed and maintained. The data stream is simply partitioned into small blocks of fixed size, then classifiers are learned from blocks. The most recent K classifiers comprise the ensemble, and old classifiers retire sequentially by age. Besides a set of training examples for classifier learning, another set of training examples are also needed for classifier weighting. If training data is sufficient, we can reserve part of it for weight training; otherwise, randomly sampled training examples can serve the purpose. We only need to make the two data sets as synchronized as possible. When sufficient training data is collected for classifier learning and ensemble weighting, the following steps are conducted: (1) learn a new classifier from the training block; (2) replace the oldest classifier in the ensemble with this newly learned; and then (3) weight the ensemble.

The rest of this section gives a formal description of ensemble weighting. A two-class classification setting is considered for simplicity, but the treatment can be extended to multi-class tasks.

The training data for ensemble weighting is represented as

$$(\mathcal{X}, \mathcal{Y}) = \{(\mathbf{x}_i, y_i); i = 1, \cdots, N\}$$

$\mathbf{x}_i$ is a vector valued sample attribute and $y_i \in \{0, 1\}$ the sample class label. We assume an ensemble of classifiers, denoted in a vector form as

$$\mathbf{f} = (f_1(\mathbf{x}), \cdots, f_K(\mathbf{x}))^T$$

where each $f_k(\mathbf{x})$ is a classifier function producing a value for the belief on a class. The individual classifiers in the ensemble may be weak or out-of-date. It is the goal of our discriminative model $\mathcal{M}$ to make the ensemble strong by weighted voting. Classifier weights are model parameters, denoted as

$$\mathbf{w} = (w_1, \cdots, w_K)^T$$

where w_k is the weight associated with classifier f_k. The model $\mathcal{M}$ also specifies for decision making a weighted voting scheme, that is,

$$\mathbf{w}^T \cdot \mathbf{f}$$

Because the ensemble prediction $\mathbf{w}^T \cdot \mathbf{f}$ is a continuous value, yet the class label y_i to be decided is discrete, a standard approach is to assume that y_i conditionally follows a Bernoulli distribution parameterized by a latent score η_i:

$$\begin{array}{l} y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w} \sim \text{Ber}(q(\eta_i)) \\ \eta_i = \mathbf{w}^{\mathbf{T}} \cdot \mathbf{f} \end{array} \tag{1}$$

where $q(\eta_i)$ is the logit transformation of η_i:

$$q(\eta_i) \triangleq \text{logit}(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}}$$

Eq.1 states that y_i follows a Bernoulli distribution with parameter q, thus the posterior probability is

$$p(y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w}) = q^{y_i}(1 - q)^{1 - y_i} \tag{2}$$

The above description leads to optimizing classifier weights using logistic regression. Given a data set $(\mathcal{X}, \mathcal{Y})$ and an ensemble $\mathbf{f}$, the logistic regression technique optimizes the classifier weights by maximizing the likelihood of the data. The optimization problem has a closed-form solution which can be quickly solved. We postpone the detailed model computation till section 4.

Logistic regression is a well-established regression method, widely used in traditional areas when the regressors are continuous and the responses are discrete [9]. In our work, we formulate the classifier weighting problem as an optimization problem and solve it using logistic regression. In section 5 we show that such a formulation and solution provide much better adaptability than previous work. (Refer to Fig.1-2, section 5 for a quick reference.)

3. Robustness to Outliers

Regression is adaptive because it always tries to fit the data from the current concept. But, it can potentially overfit outliers. We integrate the following outlier detection technique into the model learning.

We define outliers as samples with a small likelihood under a given data model. The goal of learning is to compute a model that best fits the bulk of the data, that is, the inliers. Whether a sample is an outlier is hidden information in this problem. This suggest us to solve the problem under the EM framework, using a robust statistical formulation.

Previously we have described a training data set as $\{(\mathbf{x}_i, y_i), i = 1, \cdots, N\}$, or $(\mathcal{X}, \mathcal{Y})$. This is an *incomplete* data set, as the outlier information is missing. A *complete* data set is a triplet

$$(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$$

where

$$\mathcal{Z} = \{z_1, \cdots, z_N\}$$

is a hidden variable that distinguishes the outliers from the inliers. $z_i = 1$ if $(\mathbf{x}_i, y_i)$ is an outlier, $z_i = 0$ otherwise. This $\mathcal{Z}$ is not observable and needs to be inferred. After the values of $\mathcal{Z}$ are inferred, $(\mathcal{X}, \mathcal{Y})$ can be partitioned into a clean sample set

$$(\mathcal{X}_{\mathbf{0}}, \mathcal{Y}_{\mathbf{0}}) = \{(\mathbf{x}_i, y_i, z_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, z_i = 0\}$$

and an outlier set

$$(\mathcal{X}_\phi, \mathcal{Y}_\phi) = \{(\mathbf{x}_i, y_i, z_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, z_i = 1\}$$

It is the samples in $(\mathcal{X}_{\mathbf{0}}, \mathcal{Y}_{\mathbf{0}})$ that all come from one underlying distribution, and are used to fit the model parameters.

To infer the outlier indicator $\mathcal{Z}$, we introduce a new model parameter λ. It is a threshold value of sample likelihood. A sample is marked as an outlier if its likelihood falls below λ. This λ, together with $\mathbf{f}$ (classifier functions) and $\mathbf{w}$ (classifier weights) discussed earlier, constitutes the complete set of parameters of our discriminative model $\mathcal{M}$, denoted as $\mathcal{M}(\mathbf{x}; \mathbf{f}, \mathbf{w}, \lambda)$.

4. Model Learning

In this section, we give the model formulation followed by model computation. The symbols used are summarized in table 1.

$(\mathbf{x}_i, y_i)$	a sample, with $\mathbf{x}_i$ the sample attribute, y_i the sample class label,
$(\mathcal{X}, \mathcal{Y})$	an incomplete data set without outlier information,
$\mathcal{Z}$	a hidden variable,
$(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$	a complete data set with outlier information,
$(\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})$	a clean data set,
$(\mathcal{X}_\phi, \mathcal{Y}_\phi)$	an outlier set,
$\mathcal{M}$	the discriminative model,
$\mathbf{f}$	a vector of classifier function, a model parameter,
$\mathbf{w}$	a vector of classifier weights, a model parameter,
λ	a threshold of likelihood, a model parameter.

Table 1. Summary of symbols used

4.1. Model Formulation

Our model has a four-tuple representation $\mathcal{M}(\mathbf{x}; \mathbf{f}, \mathbf{w}, \lambda)$. Given a training data set $(\mathcal{X}, \mathcal{Y})$, an ensemble of classifiers $\mathbf{f} = (f_1(\mathbf{x}), \cdots, f_K(\mathbf{x}))^T$, we want to achieve two objectives.

1. To infer about the hidden variable $\mathcal{Z}$ that distinguishes inliers $(\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})$ from outliers $(\mathcal{X}_\phi, \mathcal{Y}_\phi)$.
2. To compute the optimal fit for model parameters $\mathbf{w}$ and λ in the discriminative model $M(\mathbf{x}; \mathbf{f}, \mathbf{w}, \lambda)$.

Each inlier sample $(\mathbf{x}_i, y_i) \in (\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})$ is assumed to be drawn from an independent identical distribution belonging to a probability family characterized by parameters $\mathbf{w}$, denoted by a density function $p((\mathbf{x}, y); \mathbf{f}, \mathbf{w})$. The problem is to find the values of $\mathbf{w}$ that maximizes the likelihood of $(\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})$ in the probability family. As customary, we use log-likelihood to simplify the computation:

$$\log\ p((\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})|\mathbf{f}, \mathbf{w})$$

A parametric model for outlier distribution is not available because outliers are highly irregular. We use instead a non-parametric statistics based on the number of outliers ($\|(\mathcal{X}_\phi, \mathcal{Y}_\phi)\|$). Then, the problem becomes an optimization problem. The score function to be maximized involves two parts: (i) the log-likelihood term for the inliers $(\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})$, and (ii) a penalty term for the outliers $(\mathcal{X}_\phi, \mathcal{Y}_\phi)$. That is:

$$\begin{aligned}(\mathbf{w}, \lambda)^* &= \arg\max_{(\mathbf{w},\lambda)} \big(\log\ p((\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})|\mathbf{f}, \mathbf{w}) \\ &\quad -\zeta((\mathcal{X}_\phi, \mathcal{Y}_\phi); \mathbf{w}, \lambda)\big) \qquad (3)\end{aligned}$$

where the penalty term, which penalizes having too many outliers, is defined as

$$\zeta((\mathcal{X}_\phi, \mathcal{Y}_\phi); \mathbf{w}, \lambda) = e \cdot \|(\mathcal{X}_\phi, \mathcal{Y}_\phi)\| \qquad (4)$$

$\mathbf{w}$ and λ affect ζ implicitly. The value of e empirically depends on the size of the training data. In our experiments we set $e \in (0.2, 0.3)$.

After expanding the log-likelihood term, we have:

$$\begin{aligned}&\log\ p((\mathcal{X}_\mathbf{0}, \mathcal{Y}_\mathbf{0})|\mathbf{f}, \mathbf{w}) \\ = &\sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{0}} \log\ p((\mathbf{x}_i, y_i)|\mathbf{f}, \mathbf{w}) \\ = &\sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{0}} \log\ p(y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w}) + \sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{0}} \log\ p(\mathbf{x}_i)\end{aligned}$$

Absorb $\sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{0}} \log\ p(\mathbf{x}_i)$ into the penalty term $\zeta((\mathcal{X}_\phi, \mathcal{Y}_\phi); \mathbf{w}, \lambda)$, and replace the likelihood in Eq.3 with the logistic form (Eq.2), then the optimization goal becomes finding the best fit $(\mathbf{w}, \lambda)^*$.

$$\begin{aligned}(\mathbf{w}, \lambda)^* &= \arg\max_{(\mathbf{w},\lambda)} \Big(\sum_{\mathbf{x}_i \in \mathcal{X}_\mathbf{0}} \big(y_i\ q + (1 - y_i)(1 - q)\big) \\ &\quad + \zeta((\mathcal{X}_\phi, \mathcal{Y}_\phi); \mathbf{w}, \lambda)\Big) \qquad (5)\end{aligned}$$

The score function to be maximized is not differentiable because of the non-parametric penalty term. We have to resort to a more elaborate technique based on the Expectation-Maximization (EM) [3] algorithm to solve the problem.

4.2. Inference and Computation

The main goal of model computation is to infer the missing variables and compute the optimal model parameters, under the EM framework. The EM in general is a method for maximizing data likelihood in problems where data is incomplete. The algorithm iteratively performs an Expectation-Step (*E-Step*) followed by an Maximization-Step (*M-Step*) until convergence. In our case,

1. E-Step: to impute / infer the outlier indicator $\mathcal{Z}$ based on the current model parameters $(\mathbf{w}, \lambda)$.
2. M-Step: to compute new values for $(\mathbf{w}, \lambda)$ that maximize the score function in Eq. 3 with current $\mathcal{Z}$.

Next we will discuss how to impute outliers in E-Step, and how to solve the maximization problem in M-Step. The M-Step is actually a Maximum Likelihood Estimation (MLE) problems.

E-Step: Impute Outliers

With the current model parameters $\mathbf{w}$ (classifier weights), the model for clean data is established as in Eq.1, that is, the class label (y_i) of a sample $\mathbf{x}_i$ follows a Bernoulli distribution parameterized with the ensemble prediction for this sample ($\mathbf{w^T} \cdot \mathbf{f}(\mathbf{x}_i)$). Thus, y_i's log-likelihood $p(y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w})$ can be computed by Eq.2.

Note that the line between outliers and inliers is drawn by λ, which is computed in the previous M-Step. So, the formulation of imputing outliers is straightforward:

$$z_i = \text{sign}\big(\log\ p(y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w}) - \lambda\big) \qquad (6)$$

where

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x < 0 \\ 0 & \text{otherwise} \end{cases}$$

M-Step: MLE

The score function (in Eq.5) to be maximized is not differentiable because of the penalty term. We consider a simple approach for an approximate solution. In this approach, the computation of λ and $\mathbf{w}$ is separated.

1. λ is computed using standard K-means clustering algorithm on log-likelihood $p(y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w})$. In our experiments we choose $K = 3$. The cluster boundaries are candidates of likelihood threshold λ^* separating outliers from inliers.

2. By fixing each of the candidate λ^*, $\mathbf{w}^*$ can be computed using the standard MLE procedure. Running a MLE procedure for each candidate λ^*, and the maximum likelihood will identify the best fit of $(\mathbf{w}, \lambda)^*$.

The standard MLE procedure for computing $\mathbf{w}$ is described as follows. Taking the derivative of the inlier likelihood with respect to $\mathbf{w}$ and set it to zero, we have

$$\frac{\partial}{\partial \mathbf{w}} \sum_{y_i \in \mathcal{Y}_o} \big(y_i \frac{e^{\eta_i}}{1 + e^{\eta_i}} + (1 - y_i) \frac{1}{1 + e^{\eta_i}}\big) = 0$$

To solve this equation, we use the Newton-Raphson procedure, which requires the first and second derivatives. For clarity of notation, we use $h(\mathbf{w})$ to denote the first derivative of inlier likelihood function with regard to $\mathbf{w}$. Starting from $\mathbf{w}_t$, a single Newton-Raphson update is

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \Big(\frac{\partial^2 h(\mathbf{w}_t)}{\partial \mathbf{w} \partial \mathbf{w}^T}\Big)^{-1} \frac{\partial h(\mathbf{w}_t)}{\partial \mathbf{w}}$$

Here we have

$$\frac{\partial h(\mathbf{w})}{\partial \mathbf{w}} = \sum_{y_i \in \mathcal{Y}_o} (y_i - q) \mathbf{f}(\mathbf{x}_i)$$

and,

$$\frac{\partial^2 h(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = - \sum_{y_i \in \mathcal{Y}_o} q(1 - q) \mathbf{f}(\mathbf{x}_i) \mathbf{f}^T(\mathbf{x}_i)$$

The initial values of $\mathbf{w}$ is important for computation convergence. Since there is no prior knowledge, we can initially set $\mathbf{w}$ to be uniform.

5. Experiments and Discussions

We use both synthetic data and a real-life application to evaluate our discriminative model's adaptability to concept shifts and robustness to noise. Our model is compared with the two previously mentioned approaches: *Bagging* [17] and *Weighted Bagging* [18]. We show that although the empirical weighting in *Weighted Bagging* performs better than unweighted voting, the robust regression weighting method is more superior, in terms of both adaptability and robustness.

C4.5 decision trees are used in our experiments, but in principle our method can be used with any base learning algorithm.

5.1. Data Sets

Synthetic Data

In the synthetic data set for controlled study, a sample $(\mathbf{x}, y)$ has three independent features $\mathbf{x} =< x_1, x_2, x_3 >$, $x_i \in [0, 1], i = 0, 1, 2$. Geometrically, samples are points in a 3-dimension unit cube. The real class boundary is a sphere defined as

$$B(\mathbf{x}) = \sum_{i=0}^{2} (x_i - c_i)^2 - r^2 = 0$$

where $\mathbf{c} =< c_1, c_2, c_3 >$ is the center of the sphere, r the radius. $y = 1$ if $B(\mathbf{x}) \le 0$, $y = 0$ otherwise. This learning task is not easy due to the continuous feature space and the non-linear class boundary.

To simulate a data stream with concept drift, we move the center $\mathbf{c}$ of the sphere that defines the class boundary between adjacent blocks. The movement is along each dimension with a step of $\pm\delta$. The value of δ controls the level of shifts from small, moderate to large, and the sign of δ is randomly assigned independently along each dimension. For example, if a block has $\mathbf{c} = (0.40, 0.60, 0.50)$, $\delta = 0.05$, the sign along each direction is $(+1, -1, -1)$, then the next block would have $\mathbf{c} = (0.45, 0.55, 0.45)$. The values of δ ought to be in a reasonable range, to keep the portion of samples that change class labels reasonable. In our setting, we consider a concept shift small if δ is around 0.02, and relatively large if δ around 0.1.

To study the model robustness, we insert noise into the training data sets by randomly flipping the class labels with a probability of p, $p = 10\%, 15\%, 20\%$. Clean testing data sets are used in all the experiments for accuracy evaluation.

Credit Card Data

The real-life application is to build a weighted ensemble for detection of fraudulent transactions in credit card transactions, data contributed by a major credit card company. A transaction has 20 features, including the transaction amount, the time of the transaction, and so on. Detailed data description is given in [16, 18]. Same as in [18], concept drift in our work is simulated by sorting transactions by the transaction amount.

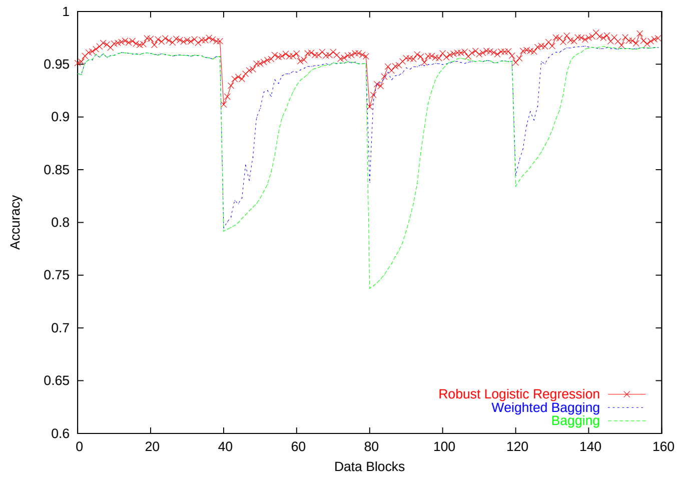

Figure 1. Adaptability comparison of the ensemble methods on data with three abrupt shifts.

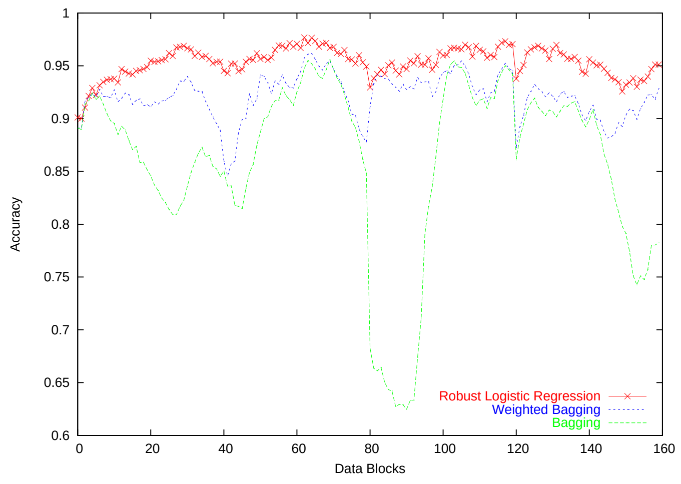

Figure 2. Adaptability comparison of the ensemble methods on data with three abrupt shifts mixed with small shifts.

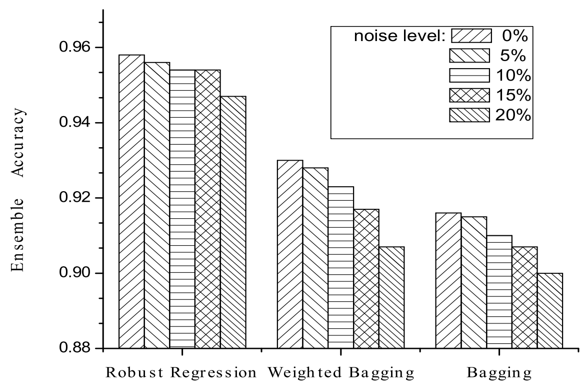

Figure 3. Robustness comparison of the three ensemble methods for different noise levels.

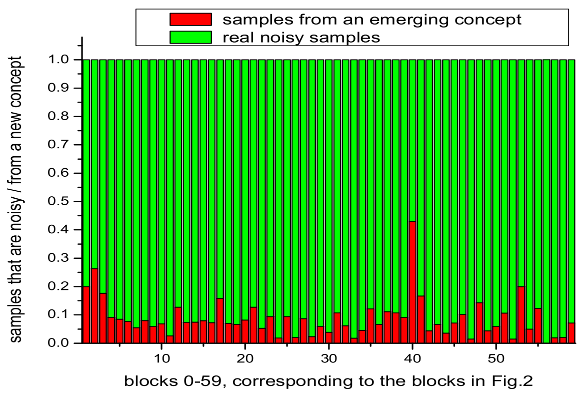

Figure 4. In the outliers detected, the normalized ratio of (1) true noisy samples (the upper bar), vs. (2) samples from an emerging concept (the lower bar). The bars correspond to blocks 0-59 in the experiments shown in Fig.2

5.2. Evaluation of Adaptation

In this subsection we compare our robust regression ensemble method with *Bagging* and *Weighted Bagging*. Concept drift is simulated by moving the class boundary center between adjacent data blocks. The moving distance δ along each dimension controls the magnitude of concept drift. We have two sets of experiments with different δ values, both have abrupt large changes occurring at block 40, 80 and 120. In one experiment, data remains stationary between these changing points. In the other experiment, small shifts are mixed between abrupt ones, with $\delta \in (0.005, 0.03)$. The percentage of positive samples fluctuates between $(41\%, 55\%)$. Noise level is 10%.

As shown in Fig.1 and Fig.2, the robust regression model always gives the best performance. The unweighted bagging ensembles has the worst predictive accuracy. Both bagging methods are seriously impaired at the concept changing points, but the robust regression is able to catch up with the new concept quickly.

5.3. Robustness in the Presence of Outliers

Noise is the major source of outliers. Fig. 3 shows the ensemble performance for the different noise levels: 0%, 5%, 10%, 15% and 20%. The accuracy is averaged over 100 runs spanning 160 blocks, with small gradual shifts between

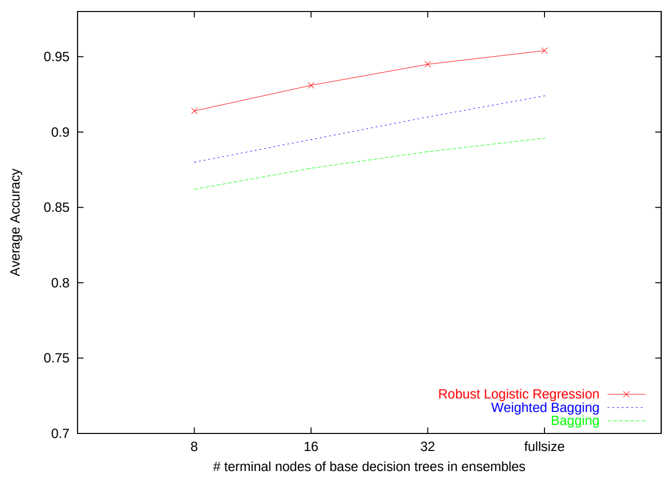

Figure 5. Performance comparison of the ensemble methods with classifiers of different size. Robust regression with smaller classifiers is compatible to the others with larger classifiers.

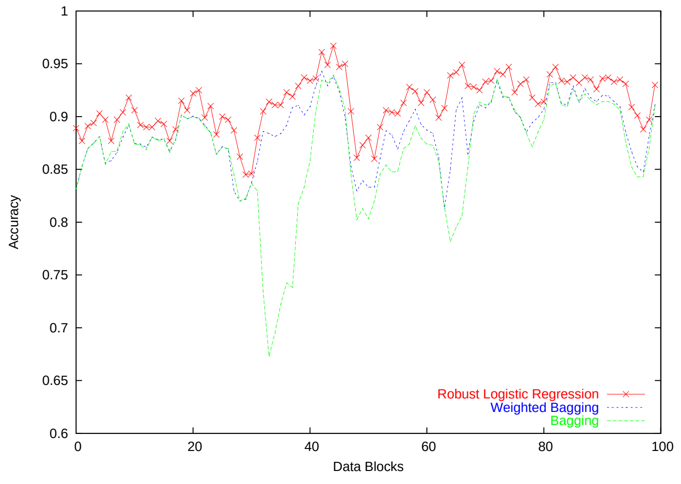

Figure 6. Performance comparison of the ensembles on credit card data. Base decision trees have no more than 16 terminal nodes. Concept shifts are simulated by sorting the transactions by the transaction amount.

blocks. We can make two major observations here:

1. The robust regression ensembles are the most accurate for all the noise levels, as clearly shown in Fig. 3.

2. Robust regression also gives the least performance drops when noise increases. This conclusion is confirmed using paired t-test at 0.05 level. In each case when noise level increases by 10%, 15% or 20%, the decrease in accuracy produced by robust regression is the smallest, and the differences are statistically significant.

To better understand why the robust regression method is less impacted by outliers, we show the outliers it detects in Fig.4. Outliers consist mostly noisy samples and samples from a newly emerged concept. In the experiments shown in Fig.2, we record the outliers in blocks 0-59 and calculate the normalized ratio of the two parts. As it shows, true noise dominates the identified outliers. At block 40 where concept drift is large, a bit more samples reflecting the new concept are mistakenly reported as outliers, but still more true noisy samples are identified at the same time.

5.4. Discussions on Performance Issue

Constrained by the requirement of on-line responses and by limited computation and memory resources, stream data mining methods should learn fast, and produce simple classifiers. For ensemble learning, simple classifiers help to achieve these goals. Here we show that simple decision trees can be used in the logistic regression model for better performance.

The simple classifiers we use are decision trees with 8, 16, 32 terminal nodes. Full grown trees are also included for comparison and denoted as “fullsize” where referred. Fig.5 compares the accuracies (averaged over 160 blocks) of the ensembles. First to note is that the robust regression method is always the best despite of the tree size. More importantly, it boosts a collection of simple classifiers, which are weak in classification capability individually, into a strong ensemble. Actually the robust regression ensemble of smaller classifiers is compatible or even better than the two bagging ensembles of larger classifiers. We observed the above mentioned superior performance of the robust regression method under different levels of noise.

For the computation time study, we verify that robust regression is compatible to weighted bagging in terms of speed. In a set of experiments where the three methods run for about 40 blocks, the learning together with evaluation time totals a 138 seconds for unweighted bagging. It is 163 seconds for weighted bagging, and 199 seconds for the robust regression. The running time is obtained when full grown decision trees are used. If small decision trees are used instead, logistic regression learning can further be sped up yet still perform better than the other two methods with full grown trees.

5.5. Experiments on Real Life Data

The real-life application is to build a classification model for detection of fraudulent transactions in credit card transactions. A transaction has 20 features including the transaction amount, the time of the transaction, etc.

We study the ensemble performance using different block size (1k, 2k, 3k and 4k), and different base models (decision trees with terminal nodes no more than 8, 16, 32 and full-size trees). We show one experiment in Fig.6, where the block size is 1k, and the base models have at most 16 terminal nodes. Results of other experiments are similar. The curve shows fewer and smaller drops in accuracy for the robust regression than for the other methods. These drops occur when the transaction amount jumps. Overall, the robust regression ensemble method performs better than the other two ensemble methods.

6. Summary and Future Work

In this paper, we propose an adaptive and robust model learning method that is highly adaptive to concept changes and is robust to noise. The model produces a weighted ensemble. The weights of classifiers are computed by logistic regression technique, which ensures good adaptability. Furthermore, this logistic regression-based weighting scheme is capable to boost a collection of weak classifiers, thus achieving the goal of fast and light learning. Outlier detection is integrated into the model learning, so that classifier weight training involves only the inliers, which leads to the robustness of the resulting ensemble. For outlier detection, we assume that the inlier's belonging to certain class follows a Bernoulli distribution, and outliers are samples with a small likelihood from this distribution. The classifier weights are estimated in a way that maximizes the training data likelihood. Compared with recent work [17, 18], the experimental results show that this statistical model achieves higher accuracy, adapts to underlying concept drift more promptly, and is less sensitive to noise.

References

[1] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *Int'l Conf. Management of Data (SIGMOD)*, 2001.

[2] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. In *Machine Learning 6(1)*, 1991.

[3] J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. In *Technical Report ICSI-TR-97-021*, 1998.

[4] L. Breiman. Bagging predictors. In *Int'l Conf. on Machine Learning (ICML)*, 1996.

[5] M. Breunig, H. Kriegel, R. Ng, and J. Sander. LOF: identifying density-based local outliers. In *Int'l Conf. Management of Data (SIGMOD)*, 2000.

[6] C. Brodley and M. Friedl. Identifying and eliminating mislabeled training instances. In *Artificial Intelligence*, 1996.

[7] C. Domeniconi and D. Gunopulos. Incremental support vector machine construction. In *Int'l Conf. Data Mining (ICDM)*, 2001.

[8] G. Dong, J. Han, V. Lakshmanan, J. Pei, H. Wang, and P. Yu. Online mining of changes from data streams: Research problems and preliminary results. In *Int'l Conf. Management of Data (SIGMOD)*, 2003.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining,Inference and Prediction*. Springer, 2000.

[10] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2001.

[11] J. Kolter and M. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Int'l Conf. Data Mining (ICDM)*, 2001.

[12] J. Kubica and A. Moore. Probabilistic nise identification and data cleaning. In *Int'l Conf. Data Mining (ICDM)*, 2003.

[13] S. Papadimitriou, H. Kitawaga, P. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *IRP-TR-02-09*, 2002.

[14] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. page Int'l Conf. Management of Data (SIGMOD), 2000.

[15] J. Schlimmer and F. Granger. Beyond incremental processing: Tracking concept drift. In *Int'l Conf. on Artificial Intelligence*, 1986.

[16] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan. Credit card fraud detection using meta-learning: Issues and initial results. In *AAAI-97 Workshop on Fraud Detection and Risk Management*, 1997.

[17] W. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2001.

[18] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2003.

[19] W. Wang, J. Yang, and P. Yu. Mining patterns in long sequential data with noise. In *SIGKDD Explorations*, 2000.

[20] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. In *Machine Learning*, 1996.