

# Mining Noisy Data Streams via a Discriminative Model

Fang Chu, Yizhou Wang, and Carlo Zaniolo

University of California, Los Angeles CA 90095, USA

**Abstract.** The two main 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 discriminative model for fast mining of noisy data streams. We build an ensemble of classifiers to achieve 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 discriminative model.

**keywords.** Data mining, discriminative modelling, robust statistics, logistic regression

## 1 Introduction

Recently there is a substantial research interest on learning data streams: the type of data that arrives continuously in high volume and speed. Applications involving stream data abound, such as network traffic monitoring, credit card fraud detection and stock market trend analysis. The work presented here focuses on the primary challenges of data stream learning: concept drift and data contamination.

Data in traditional learning tasks is stationary. That is, the underlying concept that maps the attributes to the class labels is unchanging. With data streams, however, the concept is not stable but drift with time due to changes in the environment. For example, customer purchase preferences change with season, availability of alternatives or services. Often the changes make the model learned from old data inconsistent with the new data, and updating of the model is necessary. This is generally known as *concept drift* [18].

Data contamination is a practical problem, introduced either due to unreliable data sources or during data transmission. Noise can greatly impair data learning. It is an even more severe problem in the context of data streams because it interferes with change adaptation. If an algorithm is too greedy to adapt to concept changes, it may overfit noise by mistakenly interpreting it as data from a new concept. If one is too robust to noise, it may overlook the changes and adjust slowly.

In this paper, we propose a novel discriminative model for adaptive learning of noisy data streams. The goal is to combine adaptability to concept drift and robustness to noise. The model produced by the learning process is in the form of an ensemble. Our work contributes in two aspects: (1) we formulate the classifier weighting problem as a regression problem, targeting at likelihood maximization of data from current concept; and, (2) we integrate outlier detection into the overall model learning.

## 2 Related Work

The problem of concept drift has been addressed in both machine learning and data mining communities. The first system capable of handling concept drift was STAGGER [14], which maintains a set of concept descriptions. Later on, there were the instance-based algorithm IB3 [2], which stores a set of instances and eliminates those when they become irrelevant or out-dated. The FLORA family [18] keeps a rule system in step with the hidden concept through a window of recent examples, and alters the window size according to changes in prediction accuracy and concept complexity. Although these algorithms introduced an interesting problem and proposed valuable insights, they were developed and tested only on very small datasets. It is not very clear how suitable they are for data streams which are 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. Work belonging to the first category includes Hoeffding tree [9], which grows a decision tree node by splitting an attribute only when that attribute is statistically predictive. The statistics is recorded and periodically checked to discover possible concept changes. However, many examples are needed to decide a split, hence the algorithm requires a large number of training samples to reach a fair performance. Domeniconi and Gunopulos [7] designed an incremental support vector machine algorithm for continuous learning. Experimental results suggest that SVM is not the ideal choice for fast learning on stream data.

Now we focus on ensemble-based approaches. Kolter and Maloof [10] proposed to track concept drift by an ensemble of experts. The poor experts are weighted low or discarded, and the good experts are updated using recent examples. One problem with this approach is the incremental base learner. Incremental algorithms are not readily available, and their updating is not easy without restricting assumptions that are impractical in many situations. For example, an assumption is made in [10] that the values of each attribute follow certain Gaussian distribution.

Other effective ensemble-based approaches simply partition the data stream into sequential blocks of fixed size and learn an ensemble from these blocks. Once a base model is constructed, it will never be updated with new examples. Two types of voting schemes are adopted. Street et al. [16] let their ensemble vote uniformly, while Wang et al. [17] prefer a weighted voting. These two approaches, [16] and [17], are closely related to what we will present in this paper, as our method also builds an ensemble from sequential blocks. In our comparative study, we will refer to these two methods as *Bagging* and *Weighted Bagging*, respectively.<sup>1</sup>

What is missing from the above mentioned work is a mechanism dealing with noise. Although there have been a number of off-line algorithms [1, 13, 5, 11, 6] for noise identification, or often indistinguishably called *outlier* detection, the concept drift problem creates a substantial gap between stream data and the existing techniques. In addition, we address the problem of finding a general distance metric. Popular distance func-

---

<sup>1</sup> These methods conform to the traditional bagging [4] and online bagging ensembles [12], where training samples are not weighted.

tions include the Euclidean distance or other distribution-sensitive functions, which are unable to treat categorical values.

Our work differs in two aspects from previous approaches to outlier detection. First, we tightly integrate outlier detection into the model learning process, since outliers also drift as concept drifts away. Secondly, the distance metric is derived from the classification performance of the ensemble, instead of a function defined in the data space. The adaptive model learning and outlier detection therefore mutually reinforce each other. An accurate model helps to identify the outliers. On the other hand, by correctly identifying and eliminating the outliers, a more accurate model can be computed.

In section 3 and section 4, we will describe the discriminative model with regard to adaptation and robustness, respectively. Section 5 gives the model formulation and computation. Extensive experimental results will be shown in section 6.

### 3 Adaptation to Concept Drift

Ensemble weighting is the key to fast adaptation. In this section 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 broken into small blocks, then a classifier can be learned from each block. The ensemble is comprised of the most recent  $K$  classifiers. Old classifiers retire sequentially by age. Besides training examples, evaluation examples (also within known class labels) are needed for weighting classifier. If training data is sufficient, part of it can be reserved for evaluation; otherwise, random samples of the most recent data blocks can serve the purpose. We only need to make the two data sets as synchronized as possible. When sufficient data is collected for training and evaluation, we do the following operations: (1) learn a new classifier from the training block; (2) replace the oldest classifier in the ensemble with this newly learned; and (3) use the evaluation data to weigh the ensemble.

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

The evaluation data is represented as

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

with  $\mathbf{x}_i$  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

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

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

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

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  $\eta_i$ :

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

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

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

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

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

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 5.

Logistic regression is a well-established regression method, widely used in traditional areas when the regressors are continuous and the responses are discrete [8]. In our work, we formulate the classifier weighting problem as an optimization problem and solve it using logistic regression. In section 6 we shows that such a formulation and solution provide much better adaptability for stream data mining, as compared to other weighting schemes such as the intuitive weighting based on accuracies. (Refer to Fig.1-2, section 6 for a quick reference.)

## 4 Robustness to Outliers

The reason regression is adaptive to changes is that it always tries to fit the data from the current concept. But, noise overfitting is a potential problem. Therefore, we propose the following outlier detection method as an integral part of the model learning.

We interpret outliers as samples with very small likelihoods under a given data model. The goal of learning is to compute a model that best fits the bulk of data, that is, the non-outliers. Outliers are hidden information in this problem. This suggest us to solve the problem under the EM framework, using a robust statistics formulation.

In Section 3, we have described a data set as  $\{(\mathbf{x}_i, y_i), i = 1, \dots, N\}$ , 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

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

is a hidden variable that distinguishes the outliers from the clean ones.  $z_i = 1$  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

$$(\mathcal{X}_0, \mathcal{Y}_0) = \{(\mathbf{x}_i, y_i, z_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, z_i = 0\}$$

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  $(\mathcal{X}_0, \mathcal{Y}_0)$  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  $\lambda$ . It is a threshold value of sample likelihood. A sample is marked as an outlier if its likelihood falls below  $\lambda$ . This  $\lambda$ , 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  $\mathcal{M}(\mathbf{x}; \mathbf{f}, \mathbf{w}, \lambda)$ .

## 5 Our Discriminative Model

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 with outlier information missing,
$\mathcal{Z}$	a hidden variable,
$(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$	a complete data set with outlier information,
$(\mathcal{X}_0, \mathcal{Y}_0)$	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,
$\lambda$	a threshold of likelihood, a model parameter.

**Table 1.** Summary of symbols used

## 5.1 Model Formulation

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

1. To infer about the hidden variable  $\mathcal{Z}$  that distinguishes non-outliers  $(\mathcal{X}_0, \mathcal{Y}_0)$  from outliers  $(\mathcal{X}_\phi, \mathcal{Y}_\phi)$ .
2. To compute the optimal fit for model parameters  $\mathbf{w}$  and  $\lambda$  in the discriminative model  $M(\mathbf{x}; \mathbf{f}, \mathbf{w}, \lambda)$ .

An assumption is made that each non-outlier sample  $(\mathbf{x}_i, y_i) \in (\mathcal{X}_0, \mathcal{Y}_0)$  is 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  $(\mathcal{X}_0, \mathcal{Y}_0)$  in the probability family. As customary, we use log-likelihood to simplify the computation:

$$\log p((\mathcal{X}_0, \mathcal{Y}_0) | \mathbf{f}, \mathbf{w})$$

A parametric model for outlier distribution is not available, due to the highly irregularity of the outlier data. Therefore, 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 clean data  $(\mathcal{X}_0, \mathcal{Y}_0)$ , and (ii) a penalty term for outliers  $(\mathcal{X}_\phi, \mathcal{Y}_\phi)$ . That is:

$$(\mathbf{w}, \lambda)^* = \arg \max_{(\mathbf{w}, \lambda)} (\log p((\mathcal{X}_0, \mathcal{Y}_0) | \mathbf{f}, \mathbf{w}) - \zeta((\mathcal{X}_\phi, \mathcal{Y}_\phi); \mathbf{w}, \lambda)) \quad (3)$$

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

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

$\mathbf{w}$  and  $\lambda$  affect  $\zeta$  implicitly. The value of  $e$  is empirical. If  $e$  is too large, outliers will tend to be misclassified as inliers to avoid a large penalty. If  $e$  is too small, many inliers will be treated as outliers and excluded from the clean data model fitting, leading to a seemingly good model that actually under-fits the true data. In our experiments we set  $e \in (0.2, 0.3)$ .

After expanding the log-likelihood term, we have:

$$\begin{aligned} & \log p((\mathcal{X}_0, \mathcal{Y}_0) | \mathbf{f}, \mathbf{w}) \\ &= \sum_{\mathbf{x}_i \in \mathcal{X}_0} \log p((\mathbf{x}_i, y_i) | \mathbf{f}, \mathbf{w}) \\ &= \sum_{\mathbf{x}_i \in \mathcal{X}_0} \log p(y_i | \mathbf{x}_i; \mathbf{f}, \mathbf{w}) + \sum_{\mathbf{x}_i \in \mathcal{X}_0} \log p(\mathbf{x}_i) \end{aligned}$$

After we absorb  $\sum_{\mathbf{x}_i \in \mathcal{X}_0} \log p(\mathbf{x}_i)$  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  $(\mathbf{w}, \lambda)^*$ .

$$(\mathbf{w}, \lambda)^* = \arg \max_{(\mathbf{w}, \lambda)} \left( \sum_{\mathbf{x}_i \in \mathcal{X}_0} (y_i q + (1 - y_i)(1 - q)) + \zeta((\mathcal{X}_\phi, \mathcal{Y}_\phi); \mathbf{w}, \lambda) \right) \quad (5)$$

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.

## 5.2 Model Inference and Learning

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) problem.

### 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  $\log p(y_i | \mathbf{x}_i; \mathbf{f}, \mathbf{w})$  can be computed from Eq.(2).

Note that the line between outliers and clean samples is drawn by  $\lambda$ , which is computed in the previous M-Step. So, the formulation of imputing outliers is straightforward:

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

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  $\lambda$  and  $\mathbf{w}$  is separated.

1.  $\lambda$  is computed using the standard K-means clustering algorithm on log-likelihood  $\log p(y_i|\mathbf{x}_i; \mathbf{f}, \mathbf{w})$ . The cluster boundaries are candidates of likelihood threshold  $\lambda^*$ , which separates outliers from clean data. There is a tradeoff between efficiency and accuracy when choosing the value of  $K$ . In our experiments, we set  $K = 3$ . A larger  $K$  value slows down the computation, but does not necessarily gain much on accuracy.
2. By fitting each of the candidate  $\lambda^*$ ,  $\mathbf{w}^*$  can be computed using the standard MLE procedure. Running an MLE procedure for each candidate  $\lambda^*$ , and the maximum score given by Eq.(5) 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 non-outlier likelihood with respect to  $\mathbf{w}$  and set it to zero, we have

$$\frac{\partial}{\partial \mathbf{w}} \sum_{y_i \in \mathcal{Y}_0} \left( y_i \frac{e^{\eta_i}}{1 + e^{\eta_i}} + (1 - y_i) \frac{1}{1 + e^{\eta_i}} \right) = 0$$

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 clean data likelihood function with regard to  $\mathbf{w}$ . Starting from  $\mathbf{w}_t$ , a single Newton-Raphson update is

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

Here we have

$$\frac{\partial h(\mathbf{w})}{\partial \mathbf{w}} = \sum_{y_i \in \mathcal{Y}_0} (y_i - q) \mathbf{f}(\mathbf{x}_i)$$

and,

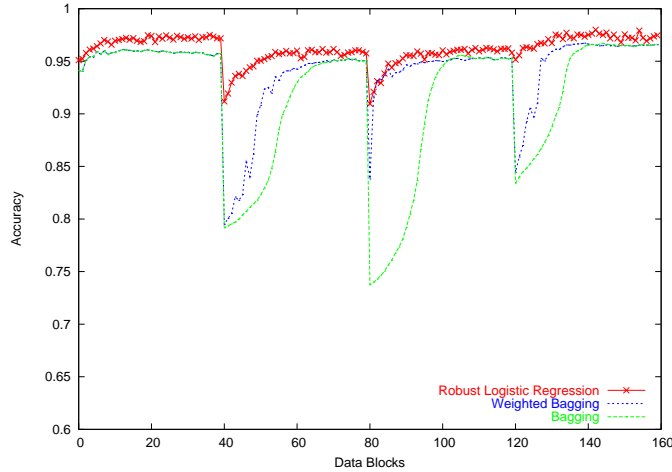
$$\frac{\partial^2 h(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = - \sum_{y_i \in \mathcal{Y}_0} q(1 - q) \mathbf{f}(\mathbf{x}_i) \mathbf{f}^T(\mathbf{x}_i)$$

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

## 6 Experiments

We use both synthetic data and a real-life application to evaluate the model's adaptability to concept shifts and robustness to noise. Our model is compared with two previous approaches: *Bagging* and *Weighted Bagging*. We show that although the empirical weighting in *Weighted Bagging* [17] 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 does not require any specific base learning algorithm.



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

## 6.1 Data Sets

### Synthetic Data

In the synthetic data set for controlled study, a sample  $(\mathbf{x}, y)$  has three independent features  $\mathbf{x} = \langle x_1, x_2, x_3 \rangle$ ,  $x_i \in [0, 1]$ ,  $i = 0, 1, 2$ . 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  $\mathbf{c} = \langle c_1, c_2, c_3 \rangle$  is the center of the sphere,  $r$  the radius.  $y = 1$  if  $B(\mathbf{x}) \leq 0$ ,  $y = 0$  otherwise. This learning task is not easy, because the feature space is continuous and the class boundary is non-linear.

To simulate a data stream with concept drift between adjacent blocks, we move the center  $\mathbf{c}$  of the sphere that defines the class boundary. The movement is along each dimension with a step of  $\pm\delta$ . The value of  $\delta$  controls the level of shifts from small, moderate to large, and the sign of  $\delta$  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  $\delta$  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  $\delta$  is around 0.02, and relatively large if  $\delta$  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  $p$ . Clean testing data sets are used in all the experiments for accuracy evaluation.

The experiments shown below are obtained when the block size equals 2k, but similar results are obtained for other block sizes (1k, 4k, 8k, etc.).

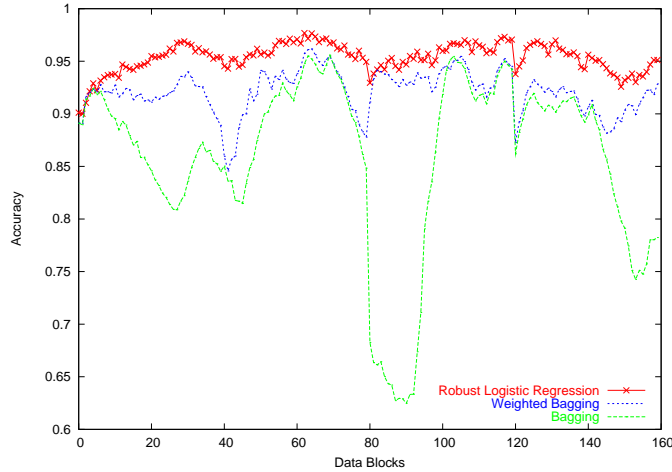


Fig. 2. Adaptability comparison of the three ensemble methods with mixed changes.

### Credit Card Data

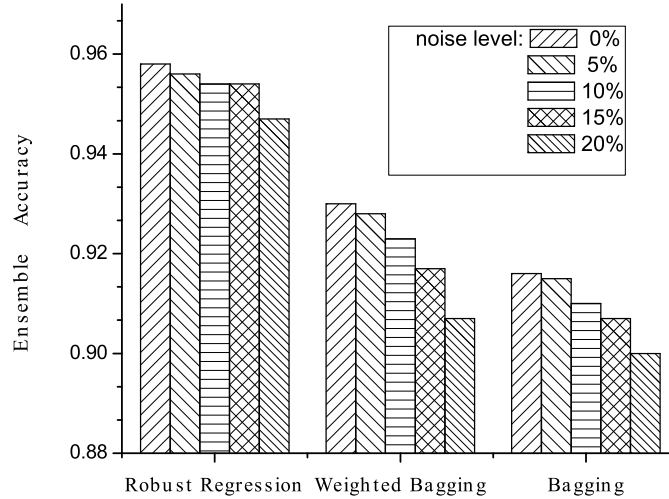
The real-life application is to build a weighted ensemble for detection of fraudulent transactions in credit card transactions, data contributed by a major credit card company. A transaction has 20 features, including the transaction amount, the time of the transaction, and so on. Detailed data description is given in [15, 17]. Same as in [17], concept drift in our work is simulated by sorting transactions by the transaction amount.

### 6.2 Evaluation of Adaptation

In this subsection we compare our robust regression ensemble method with *Bagging* and *Weighted Bagging*. Concept shifts are simulated by the movement of the class boundary center, which occur between adjacent data blocks. The moving distance  $\delta$  along each dimension controls the magnitude of concept drift. We have two sets of experiments with varying  $\delta$  values, both have abrupt changes occurring every 40 blocks, which means that the abrupt concept changing points are block 40, 80 and 120, where  $\delta$  is around 0.1. 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%).

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

In the above experiments, noises are around 10%. Experiments with varying noise levels are discussed in the next subsection. In terms of learning speed, robust regression is comparable to, although a little slower than, the bagging ensembles. For example, the total learning time on 40 blocks, each containing 2000 samples, is 76 seconds for bagging, 90 seconds for weighted bagging, and 110 seconds for robust logistic regression.



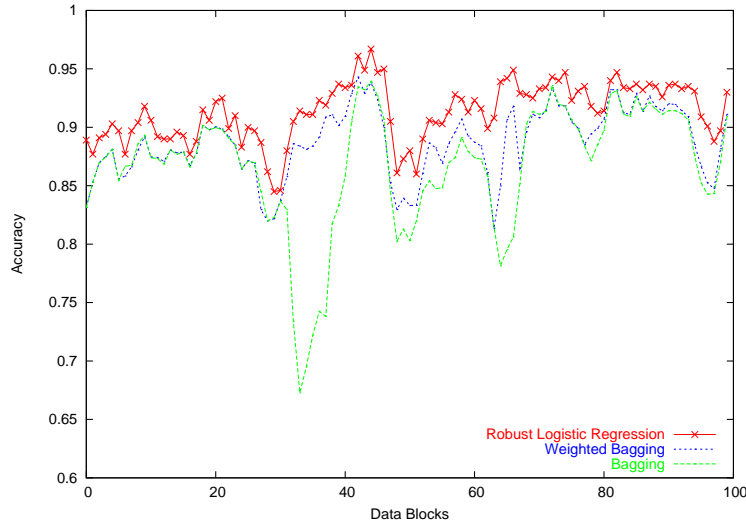
**Fig. 3.** Robustness comparison of the three ensemble methods under varying noise levels.

### 6.3 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 blocks. We can make two major observations here:

1. The robust regression ensembles are the most accurate for all the different 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.

The robustness results from the fact that the noisy samples/outliers are treated as hidden information in the model learning. Once inferred, the outliers are excluded from the clean data model fitting. The experiments on synthetic data sets indeed prove that a majority of noisy samples are captured in the learning procedure. In one of the experiments, for instance, we inserted 10% noise into a 2K data block. Our model identified 217 outliers, among which 189 were true noisy samples. The false positive rate was low. This confirms experimentally that our robust regression technique is good at separating data caused by concept drift from noisy data.



**Fig. 4.** Performance comparison of the three ensembles on credit card data. Concept drift is simulated by sorting the transactions by transaction amount.

#### 6.4 Experiments on Real Life Data

In the credit card application, we build a classification model to detection of fraudulent transactions. A transaction has 20 features including the transaction amount, the time of the transaction, etc. We study the ensemble performance using varying block size (1k, 2k, 3k and 4k). We show one experiment in Fig.4 with a block size of 1k. The curve shows fewer and smaller drops in accuracy with robust regression than with the other methods. These drops actually occur when the transaction amount jumps.

### 7 Summary

In this paper, we propose a discriminative model 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 adaptation. Furthermore, outlier detection is integrated into the model, so that classifier weight training involves only the clean data, which leads to the robustness of the resulting ensemble. For outlier detection, we assume that a clean sample's belonging to certain class follows a Bernoulli distribution, which is parameterized by the ensemble prediction. Outliers can thus be identified as samples each with a very small likelihood. The classifier weights are thus estimated to maximize the data likelihood of the clean samples.

Compared with recent works [16, 17], the experimental results show that our discriminative model achieves higher accuracy, adapts to underlying concept shifts more promptly, and is less sensitive to noise. These benefits are attributed to the classifier weighting scheme using logistic regression, and the integral outlier detection technique.

## 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)*, 37-66, 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 *Machine Learning 24(2)*, 123-140, 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*, 799-805, 1996.
7. C. Domeniconi and D. Gunopulos. Incremental support vector machine construction. In *Int'l Conf. Data Mining (ICDM)*, 2001.
8. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction*. Springer, 2000.
9. G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 2001.
10. J. Kolter and M. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. In *Int'l Conf. Data Mining (ICDM)*, 2001.
11. J. Kubica and A. Moore. Probabilistic noise identification and data cleaning. In *Int'l Conf. Data Mining (ICDM)*, 2003.
12. N.C. Oza and S. Russell. Online bagging and boosting. In *Artificial Intelligence and Statistics*, 105-112, 2001.
13. S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Int'l Conf. Management of Data (SIGMOD)*, 2000.
14. J. Schlimmer and F. Granger. Beyond incremental processing: Tracking concept drift. In *Proc. of Int'l Conf. on Artificial Intelligence 502-507*, 1986.
15. 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.
16. 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.
17. 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.
18. G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. In *Machine Learning 23*, 69-101, 1996.