Mining Concept-Drifting Data Streams Containing Labeled and Unlabeled Instances

Hanen Borchani, Pedro Larrañaga, and Concha Bielza

Departamento de Inteligencia Artificial, Facultad de Informática
Universidad Politécnica de Madrid, Boadilla del Monte, 28660, Madrid, Spain
hanen.borchani@upm.es, {pedro.larranaga,mcbielza}@fi.upm.es

Abstract. Recently, mining data streams has attracted significant attention and has been considered as a challenging task in supervised classification. Most of the existing methods dealing with this problem assume the availability of entirely labeled data streams. Unfortunately, such assumption is often violated in real-world applications given that obtaining labels is a time-consuming and expensive task, while a large amount of unlabeled instances are readily available. In this paper, we propose a new approach for handling concept-drifting data streams containing labeled and unlabeled instances. First, we use KL divergence and bootstrapping method to quantify and detect three possible kinds of drift: feature, conditional or dual. Then, if any occurs, a new classifier is learned using the EM algorithm; otherwise, the current classifier is kept unchanged. Our approach is general so that it can be applied with different classification models. Experiments performed with naive Bayes and logistic regression, on two benchmark datasets, show the good performance of our approach using only limited amounts of labeled instances.

1 Introduction

Data streams are infinite flows of highly rapid generated instances, that pose several challenges on computing systems due to limited resources of time and memory. Furthermore, as they are continuously gathered over time, data streams are characterized by their concept-drifting aspect [6,18]. That is, the learned concepts and/or the underlying data distribution are constantly evolving over time, which make the current classifier out-of-date requiring to be updated.

Recently, learning from concept-drifting data streams has become an important task for dealing with a wide range of real-world applications including network monitoring, information filtering, fraud and intrusion detection, etc. Several methods have been proposed to detect concept drift in incoming data streams and update accordingly the classifier [2,8,9,13,18,20].

However, most of these methods assume the availability of labeled data to learn accurately, and in general, they continuously monitor the classification performance over time and detect a concept drift if there is a significant fall in the performance. Unfortunately, the availability assumption of entirely labeled data streams is often violated in real-world problems as true labels may be scarce and not readily available.

Semi-supervised learning methods are useful in such cases since they use large amounts of unlabeled data, together with the labeled data, to learn better classifiers [22]. However, they mainly assume that data is generated according to some stationary distribution, which is not true when learning from evolving data streams, where changes may occur over time.

In this paper, we propose a new semi-supervised learning approach for concept-drifting data streams. Our aim is to take advantage of unlabeled data to detect possible concept drifts and update, if necessary, the classifier over time even if only a few labeled data is available.

To this end, inspired by a previous work of Dasu et al. [3], we use the Kullback-Leibler (KL) divergence [14] to measure distribution differences between data stream batches. Then, based on a bootstrapping method [5], we determine whether the obtained KL measures are statistically significant or not, i.e. whether a drift occurs or not. However, our approach differs from Dasu's work in three key points: First, we do not only detect whether a drift occurs or not, but we further distinguish and monitor three possible kinds of drift: feature, conditional or dual. Second, we do not assume the availability of entirely labeled data streams, but we detect possible drifts using labeled and unlabeled instances. Moreover, we propose a general approach to learn from these instances. In fact, when any of the three possible kinds of drift is detected, a new classifier is learned using the Expectation Maximization (EM) algorithm [4]. EM has been widely used in semi-supervised learning showing that it improves classification accuracy especially when having a small number of labeled data [17]. Otherwise, i.e. when no drift is detected, the current classifier is kept unchanged. Note that, our approach is general so that it can be applied with different classification learning algorithms. In this work, we consider two classifiers, namely naive Bayes and logistic regression.

The remainder of this paper is organized as follows. Section 2 defines the concept drift problem and briefly reviews related work. Section 3 introduces our new approach. Section 4 presents the experimental study. Finally, Section 5 rounds the paper off with some conclusions.

2 Concept Drift

2.1 Problem Definition

In dynamic environments, characteristic properties of data streams are often not stable but change over time. This is known as the concept drift problem [18,21].

Formally, concept drift is defined as the change in the joint probability distribution $P(\mathbf{x}, c)$ where c is the class label of a feature vector $\mathbf{x}$. $P(\mathbf{x}, c)$ is the product of the posterior distribution $P(c \mid \mathbf{x})$ and the feature distribution $P(\mathbf{x})$. According to [9], three possible sources of concept drifts can be distinguished as follows:

– Conditional change: In this case, a change in $P(c \mid \mathbf{x})$ occurs, that is, the class labels change given the feature vectors. For instance, in information

filtering domain consisting in classifying a stream of documents, denoted by $\mathbf{x}$, into relevant or non relevant, denoted by c, a conditional change may occur if the relevance of some documents changes over time, that is, their class labels change from relevant to non relevant or vice versa.
- Feature change: In this case, a change in $P(\mathbf{x})$ occurs. Intuitively, some previously infrequent feature vectors may become more frequent or vice versa. For instance, in information filtering domain, the relative frequency of some documents changes over time independently of their relevance, which may remain constant over a long period of time.
- Dual change: In this case, both changes in $P(\mathbf{x})$ and $P(c \mid \mathbf{x})$ occur. According to information filtering example, both changes in the relative frequency and the relevance of some documents are observed.

Note that, when any of these three kinds of concept drift occurs, the current classifier becomes out-of-date and a decrease in the accuracy is usually observed.

2.2 Related Work

Different methods have been proposed to handle concept drifting data streams. As pointed out in [8], these methods can be classified into *blind methods* (e.g. weighted examples [11], time window of fixed size [20,21]) that adapt the classifier at regular intervals without considering whether changes have really occurred, and *informed methods* (e.g. time window of adaptive size [21]) that only adapt the classifier whenever a change is detected.

Clearly, informed methods are more interesting since they present a more efficient way to cope with concept drifts and avoid the non controlled updating of the current classifier. The main issue is how to detect concept drifts. Most of existing works monitor, at least, one performance indicator over time [2,8,21]. The classification accuracy is the most used indicator, i.e. if there is a consistent drop in the accuracy, a drift is signaled. Other performance indicators such as the error rate, recall and precision have also been used.

An alternative approach to detect the drift is to monitor the data distribution on two different windows [3,10,19]. It is assumed that as long as the distribution of old instances is similar to the distribution of recent ones, no concept drift occurred. Otherwise, a distribution difference indicates a concept drift.

Contrary to previously proposed methods, our approach differs in the following ways: First, we do not only assert the presence or the absence of drift but we also determine efficiently which kind of drift has occurred (i.e. feature, conditional or dual) and quantify it. Second, we do not assume the availability of entirely labeled data to deal accurately with evolving data streams, but we tackle through this work a more realistic and important problem of mining data streams when only a small amount of labeled data is available.

To the best of our knowledge, only two relevant previous works have addressed the problem of scarceness of labeled instances in data streams. The first one [12] is based on transductive support vector machines (TSVMs) and it maintains two separate adaptive windows on labeled and unlabeled data in order to monitor, respectively, the probabilities $P(c \mid \mathbf{x})$ and $P(\mathbf{x})$ that may change at different rates. The second work was recently proposed by [15]. It is based on an ensemble approach where each model in the ensemble is built as micro-clusters using a semi-supervised clustering technique. To cope with stream evolution, the ensemble is refined via adding a new model and removing the worst one.

3 Mining Data Streams with Partially Labeled Data

Let D denote the data stream that arrives over time in batches. Let D^s denotes the batch at step s, which consists of the union of two disjoint subsets D^s_u and D^s_l. D^s_u denotes a set of N^s_u unlabeled instances $(\mathbf{x})$, while D^s_l denotes a set of N^s_l labeled instances $(\mathbf{x}, c)$, s.t. $\mathbf{x}$ represents an n-dimensional feature vector $(x_1, ..., x_n)$ and $c \in \Omega_C = \{c_1, c_2, ..., c_{|C|}\}$ represents the corresponding class value for labeled instances. $N^s = N^s_u + N^s_l$ denotes the total size of D^s.

3.1 Classifier Learning

Learning a classifier from data D^s corresponds to maximizing the joint log likelihood of D^s given the parameters $\boldsymbol{\theta}^s$, expressed as follows [17]:

$$\begin{aligned} LL(D^s \mid \boldsymbol{\theta}^s) = & \sum_{i=1}^{N^s_l} log(P(c_j \mid \mathbf{x}_i; \boldsymbol{\theta}^s) P(\mathbf{x}_i \mid \boldsymbol{\theta}^s)) \\ & + \sum_{i=1}^{N^s_u} log \sum_{j=1}^{|C|} P(c_j \mid \mathbf{x}_i; \boldsymbol{\theta}^s) P(\mathbf{x}_i \mid \boldsymbol{\theta}^s) \end{aligned} \tag{1}$$

where the first term is derived from labeled instances and the second one is based on unlabeled data. Notice that this equation contains a log of sums for the unlabeled data, which makes a maximization by partial derivatives with respect to $\boldsymbol{\theta}^s$ analytically intractable.

Consider that we can have access to the class labels of all the instances, represented using a matrix of binary indicator variables $\mathbf{z}$, where rows correspond to different instances and columns to different classes, so that an entry is $z_{ij} = 1$ iff c_j is the class of instance $\mathbf{x}_i$ and $z_{ij} = 0$ otherwise. Thus, (1) can be rewritten as follows:

$$LL(D^s \mid \boldsymbol{\theta}^s; \mathbf{z}) = \sum_{i=1}^{N^s} \sum_{j=1}^{|C|} z_{ij} \; log(P(c_j \mid \mathbf{x}_i; \boldsymbol{\theta}^s) P(\mathbf{x}_i \mid \boldsymbol{\theta}^s)) \tag{2}$$

We use the EM algorithm [4] to find the maximum $\hat{\boldsymbol{\theta}}^s$ of (2). Let $\hat{\mathbf{z}}_t$ and $\hat{\boldsymbol{\theta}}^s_t$ denote the estimates for $\mathbf{z}$ and $\boldsymbol{\theta}^s$ at iteration t. EM starts with an initial estimate of classifier parameters $\hat{\boldsymbol{\theta}}^s_1$ from only the labeled data D^s_l. Then, it iterates over the E- and M-steps:

– The E-step: uses the current classifier parameters to probabilistically label the unlabeled instances in D_u^s. Formally, it computes the expected value of:

$$\hat{\mathbf{z}}_{t+1} = E[\mathbf{z} \mid D^s; \hat{\boldsymbol{\theta}}_t^s] \tag{3}$$

Clearly, for labeled data z_{ij} is easily determined since classes are already known. For unlabeled data, z_{ij} should be estimated as follows:

$$E[z_{ij} \mid D^s; \hat{\boldsymbol{\theta}}_t^s] = \begin{cases} 1 & \text{if } c_j = argmax_c P(c \mid \mathbf{x}_i; \hat{\boldsymbol{\theta}}_t^s) \\ 0 & \text{otherwise} \end{cases}$$

– The M-step: re-estimates the classifier for the whole data D^s, i.e. using all instances, the originally labeled and the newly labeled by the E-step. In fact, this step consists in computing new parameters $\hat{\boldsymbol{\theta}}_{t+1}^s$ using the current expected value of $\hat{\mathbf{z}}$. Formally, we have:

$$\hat{\boldsymbol{\theta}}_{t+1}^s = argmax_{\boldsymbol{\theta}^s} LL(D^s \mid \boldsymbol{\theta}^s; \hat{\mathbf{z}}_{t+1}) \tag{4}$$

These two steps are iterated until reaching convergence as proved by Dempster et al. [4]. At convergence, EM finds $\hat{\boldsymbol{\theta}}^s$ that locally maximizes the log likelihood with respect to both labeled and unlabeled data.

We consider in this work two different classifiers:

- Naive Bayes (NB) [16]: For NB, parameters $\boldsymbol{\theta}^s$ denote the probability tables of class C and of each feature variable given C.
- Logistic regression (LR) [7]: For LR, the conditional log likelihood is maximized instead of the log likelihood. Hence, the expression $P(\mathbf{x}_i \mid \boldsymbol{\theta}^s)$ is removed from both equations (1) and (2). Moreover, parameters $\boldsymbol{\theta}^s$ are represented by the vector $(\boldsymbol{\theta}_{j0}^s, \boldsymbol{\theta}_{j1}^s, .., \boldsymbol{\theta}_{jn}^s)^T$ for $j = 1, ..., |C|$.

3.2 Detecting Concept Drift

Given a new batch of data D^{s+1}, the objective is to detect changes whenever they occur and adapt if necessary the current classifier. It is assumed that as long as the joint probability distribution of D^{s+1} is similar to the distribution of D^s, no concept drift occurs. Otherwise, a concept drift should be indicated.

Thus, in order to detect possible changes, we use the KL divergence [14] to measure differences between the empirical distributions of D^{s+1} and D^s. Note that the KL divergence has two fundamental properties, namely, the non-negativity, being 0 iff the two compared distributions are the same, and its asymmetry. Moreover, a higher KL value indicates a higher dissimilarity between distributions, and so, a pronounced drift.

First, using only the labeled part of D^{s+1} and D^s, we proceed to measure the KL divergence between the conditional distributions of the class given the features, in order to monitor the conditional change:

$$kl_{cc} = KL(\hat{P}_{D_l^{s+1}}(C \mid \mathbf{x}) || \hat{P}_{D_l^s}(C \mid \mathbf{x})) = \sum_{j=1}^{|C|} \hat{P}_{D_l^{s+1}}(c_j \mid \mathbf{x}) log_2 \frac{\hat{P}_{D_l^{s+1}}(c_j \mid \mathbf{x})}{\hat{P}_{D_l^s}(c_j \mid \mathbf{x})} \tag{5}$$

In addition, using all the data (labeled and unlabeled), except the class variable, we measure the KL divergence between the feature distributions, to monitor the feature change:

$$kl_{fc} = KL(\hat{P}_{D^{s+1}}(\mathbf{x})||\hat{P}_{D^s}(\mathbf{x})) = \sum_{\mathbf{x}} \hat{P}_{D^{s+1}}(\mathbf{x}) log_2 \frac{\hat{P}_{D^{s+1}}(\mathbf{x})}{\hat{P}_{D^s}(\mathbf{x})} \tag{6}$$

In order to determine whether the computed KL measures are statistically significant or not, we use the bootstrapping method [5] following the previous work in [3]. Specifically, to decide whether the obtained kl_{cc} value is significant or not, we consider the null hypothesis: $H_{0_{cc}} : P_{D_l^{s+1}}(C \mid \mathbf{x}) = P_{D_l^s}(C \mid \mathbf{x})$ denoting that no conditional change has occurred. So, our objective is to determine the probability of observing the value kl_{cc} if $H_{0_{cc}}$ is true.

To this end, given the empirical distribution $\hat{P}_{D_l^s}(C \mid \mathbf{x})$, we sample k data sets denoted S_b, $b = 1, ..., k$, each of size $2N_l^s$. Then, we consider the first N_l^s elements S_{b1} as coming from the distribution $\hat{P}_{D_l^s}(C \mid \mathbf{x})$, and the remaining N_l^s elements $S_{b2} = S_b \setminus S_{b1}$ as coming from the other distribution $\hat{P}_{D_l^{s+1}}(C \mid \mathbf{x})$; and we compute bootstrap estimates $\hat{kl}_{ccb} = KL(S_{b2}||S_{b1})$ between the two samples S_{b2} and S_{b1}. These estimates form an empirical distribution from which we construct a critical region $[\hat{kl}_{cc}^{\alpha}, \infty)$, where $\hat{kl}_{cc}^{\alpha}$ represents the $(1 - \alpha)$-percentile of the bootstrap estimates, and α is a desired significance level.

Finally, if we observe that kl_{cc} falls into the critical region, i.e. $kl_{cc} > \hat{kl}_{cc}^{\alpha}$, we conclude that it is statistically significant and invalidates $H_{0_{cc}}$. In other words, we conclude that a conditional change is detected.

Similarly, in order to decide whether the obtained kl_{fc} value is significant or not, we consider the null hypothesis: $H_{0_{fc}} : P_{D^{s+1}}(\mathbf{x}) = P_{D^s}(\mathbf{x})$ and apply the same process to determine the critical region $[\hat{kl}_{fc}^{\alpha}; \infty)$ and decide about a feature change. Note that, in case that either a feature or conditional change is detected, we proceed to learn a new classifier. Otherwise, we keep the current classifier unchanged. Algorithm 1 outlines the whole proposed approach.

4 Experimental Study

For experiments, we consider the benchmark synthetic data set on a rotating hyperplane, widely used to simulate the concept drift problem [6,9,18,20]. A synthetic data allows us to carry out experiments with different types of drift as well as different percentages of labeled data and, hence, to investigate the performance of our approach under controlled situations.

A hyperplane in d-dimensional data is denoted by $\sum_{i=1}^{n} w_i x_i = w_0$ where $\mathbf{w} = (w_1, ..., w_d)^T$ is the weight vector. Instances for which $\sum_{i=1}^{n} w_i x_i \geq w_0$ are labeled positive, and those for which $\sum_{i=1}^{n} w_i x_i < w_0$ are labeled negative. Weights w_i are initialized by random values in the range of $[0, 1]$, and w_0 values are determined so that $w_0 = \frac{1}{2} \sum_{i=1}^{n} w_i$. We generated x_i from a Gaussian distribution with mean μ_i and variance σ_i^2. The feature change is simulated

Algorithm 1

```
Input : $D^s, \boldsymbol{\theta}^s, D^{s+1}, k, \alpha$
Output : $\boldsymbol{\theta}^{s+1}$
Compute $kl_{cc}$
Compute the bootstrap estimates $\hat{kl}_{ccb}$, $b = 1, ..., k$, and critical region $[\hat{kl}^{\alpha}_{cc}, \infty)$
Compute $kl_{fc}$
Compute the bootstrap estimates $\hat{kl}_{fcb}$, $b = 1, ..., k$, and critical region $[\hat{kl}^{\alpha}_{fc}, \infty)$
if $kl_{cc} > \hat{kl}^{\alpha}_{cc}$ or $kl_{fc} > \hat{kl}^{\alpha}_{fc}$ then
    A change is detected, learn a new classifier from $D^{s+1}$
    $\hat{\boldsymbol{\theta}}^{s+1}_1 \leftarrow$ initial parameters induced only from labeled data $D^{s+1}_l$
    while not convergence do
        E-step: compute the expected labels for all unlabeled instances using (3)
        M-step: update classifier parameters using (4) obtaining $\hat{\boldsymbol{\theta}}^{s+1}$
    end while
    $\boldsymbol{\theta}^{s+1} \longleftarrow \hat{\boldsymbol{\theta}}^{s+1}$
else
    No change is detected: $\boldsymbol{\theta}^{s+1} \longleftarrow \boldsymbol{\theta}^s$
end if
return $\boldsymbol{\theta}^{s+1}$
```

through the change of the mean, i.e. μ_i is changed to $\mu_i s_i(1+t)$, and the conditional change is simulated by the change of weights w_i to $w_i s_i(1+t)$. Parameter $t \in [0, 1]$ represents the magnitude of changes, and parameter $s_i \in \{-1, 1\}$ specifies the direction of changes which could be reversed with a probability of 0.1. We generated a data stream of 10 dimensions ($n = 10$) with 80,000 instances, each block of 20,000 using a different magnitude of change t which is respectively set to 0.1, 0.2, 0.5, 1. We consider equal training and testing subsets of size 1000, such that every training set is followed by a testing set. Also, we use $k = 500$ and $\alpha = 0.05$ as bootstrap parameter values.

Table 1 represents the obtained results for the drift detection proposal. The first column represents the block numbers. Then, in columns 2 and 3, the kl_{fc} and average $\hat{kl}^{\alpha}_{fc}$ values, which are the same for all experiments with different percentages of labeled data, since they only use the features values. Finally, in columns 4 to 9, kl_{cc} and $\hat{kl}^{\alpha}_{cc}$ values respectively for 2%, 5% and 10% of labeled data. Due to lack of space, details for kl values computed over sequential blocks having the same magnitude of change t (e.g. from block 1 to block 10) are omitted. Instead, the average of these kl values are included. Note that, neither feature changes nor conditional changes are detected between these blocks.

As expected, a feature change is only detected between blocks 10 and 11 where the magnitude of change t goes from 0.1 to 0.2, blocks 20 and 21 where t goes from 0.2 to 0.5, and blocks 30 and 31 where the t goes from 0.5 to 1. The larger the modification of t values, the higher are the kl_{fc} values, showing a more significant drift in the feature distributions between considered data blocks.

The same is observed for the conditional distributions monitored via kl_{cc} values, for both 5% and 10% of labeled data. However, in the case of 2% of labeled data conditional changes are not detected. This can be explained by the

Table 1. Drift detection results for rotating hyperplane dataset

			2% labeled		5% labeled		10% labeled	
blocks	kl_{fc}	$\hat{kl}^{\alpha}_{fc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$
1 − 10	0.1132	0.1431	0.7193	4.9660	0.4007	1.2400	0.2087	0.5845
10 − 11	**0.1434**	**0.1405**	3.2875	6.4493	**1.3862**	**1.2369**	**0.9812**	**0.6499**
11 − 20	0.1176	0.1449	1.6984	6.5568	0.8258	1.6646	0.1873	0.5800
20 − 21	**0.1544**	**0.1408**	4.1283	6.4512	**1.3821**	**1.2364**	**1.0118**	**0.8162**
21 − 30	0.1063	0.1413	1.2063	6.4298	0.4126	1.3954	0.1544	0.3882
30 − 31	**0.1767**	**0.1466**	4.7233	6.4346	**1.9310**	**1.3660**	**1.3369**	**0.7648**
31 − 40	0.1164	0.1429	0.8804	5.7063	0.4345	1.0492	0.1962	0.5035

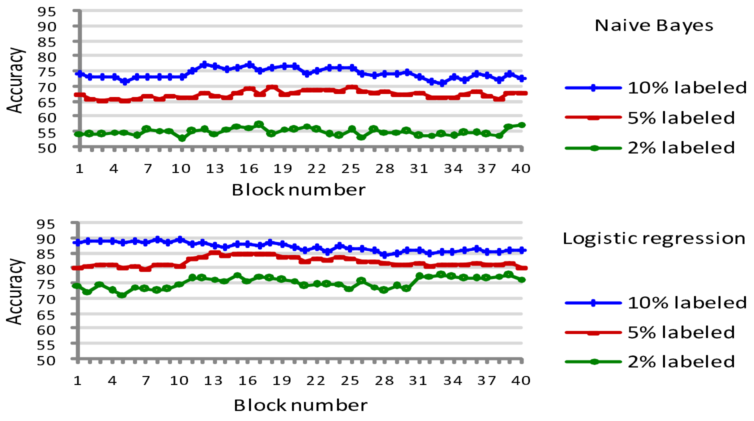

Fig. 1. Classification results of NB and LR for rotating hyperplane dataset

fact that the true conditional distribution can not be accurately approximated with very few labeled instances.

Furthermore, Figure 1 presents accuracy curves for NB and LR. Obviously, the performance of both NB and LR is much better when higher percentages of labeled data are considered. Note also that LR always outperforms NB for all percentages of labeled data.

As additional experiments, we consider the real mushroom dataset from the UCI repository [1] regarded as having virtual concept drift (i.e. feature changes) but no real concept drift (i.e. conditional changes) [13]. Mushroom dataset contains 22 variables and 8124 instances. We split it into 6 blocks, from each block 1000 instances are used for training and 354 instances for testing. As before, we use $k = 500$ and $\alpha = 0.05$ as bootstrap parameter values, and we consider three percentages of randomly selected labeled data: 2%, 5% and 10%.

According to results in Table 2, only feature changes are detected between blocks 1 and 2, and blocks 4 and 5. However, no conditional changes are detected for all percentages of labeled data, which proves that our detection method is resilient to false alarms. Moreover, according to Figure 2, using more labeled

Table 2. Drift detection results for mushroom dataset

			2% labeled		5% labeled		10% labeled	
blocks	kl_{fc}	$\hat{kl}^{\alpha}_{fc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$	kl_{cc}	$\hat{kl}^{\alpha}_{cc}$
$1-2$	**0.2251**	**0.0450**	0.0003	0.1430	0.0001	0.0035	0.0079	0.0176
$2-3$	0.0365	0.0755	0.0019	0.0156	0.0005	0.0030	0.0003	0.0009
$3-4$	0.1184	0.1536	0.0031	0.0049	0.0054	0.0057	0.0002	0.0181
$4-5$	**1.2973**	**0.2373**	0.0043	0.1347	0.0002	0.0063	0.0013	0.0030
$5-6$	0.0007	0.0814	0.0028	0.0105	0.0018	0.0092	0.0001	0.0053

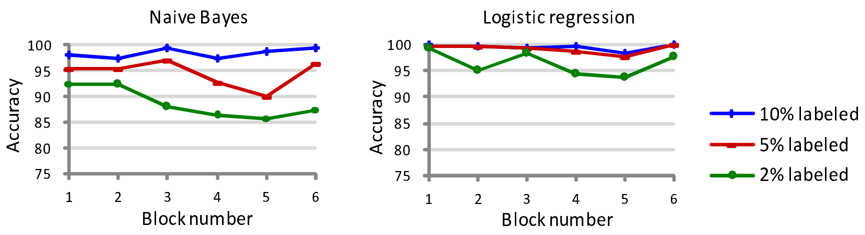

Fig. 2. Classification results of NB and LR for mushroom dataset

data improves the predictive accuracies of both NB and LR. Nevertheless, for LR, from 5% to 10% of labeled data, the improvement is very small, so that the corresponding curves are almost superimposed. Notice also that, LR always outperforms NB for mushroom dataset.

5 Conclusion

We deal with an important problem in data stream mining which has not been addressed by the most existing works assuming that data streams are entirely labeled. In our work, using both labeled and unlabeled instances, we distinguish and detect three possible changes: feature, conditional or dual, using KL divergence and bootstrapping method, then, if required, we update the classifier using EM algorithm. Experiments with naive Bayes and logistic regression show the effectiveness and the good performance of our approach. For the future, we plan to carry out additional experiments especially on more complex data streams.

Acknowledgement

This work has been supported by the Spanish Ministry of Science and Innovation under projects TIN2007-62626, TIN2008-06815-C02-02, Consolider Ingenio 2010-CSD2007-00018, and the Cajal Blue Brain project.

References

1. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, University of California, Irvine (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

2. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: Proc. of the 7th SIAM Int. Conf. on Data Mining, pp. 29–40 (2007)
3. Dasu, T., Krishnan, S., Venkatasubramanian, S., Yi, K.: An information-theoretic approach to detecting changes in multi-dimensional data streams. In: Proc. of the 38th Symposium on the Interface of Statistics, Computing Science, and Applications, pp. 1–24 (2006)
4. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society 39, 1–38 (1977)
5. Efron, B., Tibshirani, R.: Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Statistical Science 1(1), 54–75 (1986)
6. Hulten, G., Spencer, L., Domingos, P.: Mining time changing data streams. In: Proc. of the 7th Int. Conf. on Knowledge Discovery and Data mining, pp. 97–106 (2001)
7. Hosmer, D.W., Lemeshow, S.: Applied Logistic Regression, 2nd edn. John Wiley & Sons, New York (2000)
8. Gama, J., Castillo, G.: Learning with local drift detection. In: Proc. of the 2nd Int. Conf. on Advanced Data Mining and Applications, pp. 42–55 (2006)
9. Gao, J., Ding, B., Fan, W., Han, J., Yu, P.S.: Classifying data streams with skewed class distributions and concept drifts. IEEE Internet Computing 12(6), 37–49 (2008)
10. Kifer, D., Ben-David, S., Gehrke, J.: Detecting change in data streams. In: Proc. of the 30th Int. Conf. on Very Large Data Bases, vol. 30, pp. 180–191 (2004)
11. Klinkenberg, R.: Learning drifting concepts: Example selection vs. example weighting. Intelligent Data Analysis 8(3), 281–300 (2004)
12. Klinkenberg, R.: Using labeled and unlabeled data to learn drifting concepts. In: Proc. of Int. Joint Conf. on Artificial Intelligence, pp. 16–24 (2001)
13. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research 8, 2755–2790 (2007)
14. Kullback, S., Leibler, R.A.: On information and sufficiency. The Annals of Mathematical Statistics 22(1), 79–86 (1951)
15. Masud, M.M., Gao, J., Khan, L., Han, J., Thuraisingham, B.: A practical approach to classify evolving data streams: Training with limited amount of labeled data. In: The 8th IEEE Int. Conf. on Data Mining, pp. 929–934 (2008)
16. Minsky, M.: Steps towards artificial intelligence. Computers and Thought, 406–450 (1961)
17. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning 39(2/3), 103–134 (2000)
18. Tsymbal, A., Pechenizkiy, M., Cunningham, P., Puuronen, S.: Dynamic integration of classifiers for handling concept drift. Information Fusion 9(1), 56–68 (2008)
19. Vorburg, P., Bernstein, A.: Entropy-based concept shift detection. In: Proc. of the 6th Int. Conf. on Data Mining, pp. 1113–1118 (2006)
20. Wang, H., Fan, W., Yu, P., Han, J.: Mining concept drifting data streams using ensemble classifiers. In: Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 226–235 (2003)
21. Widmer, G., Kubat, M.: Learning in the presence of concept drift and hidden contexts. Machine Learning 23(1), 69–101 (1996)
22. Zhu, X.: Semi-supervised learning literature survey. Technical Report, Computer Sciences, University of Wisconsin-Madison (2008)