Pattern Recognition 43 (2010) 3151–3161

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Transfer estimation of evolving class priors in data stream classification

Zhihao Zhang, Jie Zhou *

Tsinghua National Laboratory for Information Science and Technology (TNList), State Key Laboratory on Intelligent Technology and Systems, Department of Automation, Tsinghua University, Beijing 100084, PR China

ARTICLE INFO

Article history:
Received 29 June 2009
Received in revised form
15 February 2010
Accepted 25 March 2010

Keywords:
Concept drift
Transfer learning
Prior estimation

ABSTRACT

Data stream classification is a hot topic in data mining research. The great challenge is that the class priors may evolve along the data sequence. Algorithms have been proposed to estimate the dynamic class priors and adjust the classifier accordingly. However, the existing algorithms do not perform well on prior estimation due to the lack of samples from the target distribution. Sample size has great effects in parameter estimation and small-sample effects greatly contaminate the estimation performance. In this paper, we propose a novel parameter estimation method called *transfer estimation*. Transfer estimation makes use of samples not only from the target distribution but also from similar distributions. We apply this new estimation method to the existing algorithms and obtain an improved algorithm. Experiments on both synthetic and real data sets show that the improved algorithm outperforms the existing algorithms on both class prior estimation and classification.

1. Introduction

Data stream classification is a hot topic in data mining research [1,2]. It has gained a high attraction due to the important applications in lots of fields, including financial applications, vehicle monitoring, sensor networks, etc. A great challenge for data stream classification is the varying class priors along the data sequence, i.e. the class priors may not be constant. A typical example is vehicle classification. We want to classify vehicles into different types, for example cars and trucks. However, the ratios of vehicles evolve along the time (for instance usually more cars in the daytime and more trucks during night). Classifiers may perform badly when the “assumed” priors of training data are different from the “true” priors of test data. This phenomenon on static data set is called biased class distribution or class imbalance [3–6]. For data streams, the class prior evolution is called virtual concept drift [7], which is mentioned as “concept drift” for convenience in the following text.

Several algorithms have been proposed to deal with concept drift problems [8–13]. Most of the existing algorithms worked in an inductive way. They first train an initial classifier on the training data and then use it to predict the labels of test samples. When the performance deteriorates, they need new labeled samples to retrain the classifier. In real-world applications, labeled samples are often difficult, expensive, or time consuming to obtain, as they require the efforts of experienced human annotators [14]. This problem can be solved by transductive learning. Transductive learning was proposed in Ref. [15], and some new interpretations include [16,17]. Different from inductive learning, transductive learning aims to generate an adjusted classifier for the given test samples. The most straightforward way of adjusting the classifier is to estimate the class priors of test samples and adjust the classifier according to Bayes’ rule. Under this framework, the key issue is how to estimate the class priors.

Class prior estimation is an important problem in the fields of information theory and statistics, which is called prior probabilities estimation (PPE) [18–20]. Confusion matrix method is one solution for PPE, which is based on the computation of confusion matrix [21]. It first estimates the confusion matrix on the training set from cross-tabulated classification frequencies provided by the classifier. Then the confusion matrix is used to infer the priori probabilities by solving a system of n linear equations with respect to the estimated class priors. In a recent study, Forman [22] focused on estimating the class priors using an improved form of the commonly used confusion matrix technique. A better prior estimation method is proposed in Refs. [23,24]. This work can be seen as a semi-supervised transductive algorithm. It uses the batch EM(BEM) algorithm to iteratively estimate the class priors on the test set. The converged estimation is used to adjust the classifier. It was proved that the EM learning in BEM algorithm is globally convergent and the (unique) maximum likelihood estimate can be found [25]. It was shown that BEM algorithm is more effective than confusion matrix method [23,24]. These prior estimation methods were all designed for static classification problems and are not suitable for data streams with concept drift, since class priors evolve along the data sequence and should be estimated dynamically. To our best knowledge, the only

* Corresponding author.
E-mail addresses: zhangzh06@mails.tsinghua.edu.cn (Z. Zhang), jzhou@tsinghua.edu.cn (J. Zhou).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.03.021

transductive algorithm that handles concept drift is the OEM algorithm proposed in Ref. [26].

OEM algorithm is an extension of BEM algorithm. When a new test sample comes, OEM algorithm gives an estimate of current class priors based on EM algorithm. The class priors are updated under exponential forgetting rule. Then the classifier is adjusted according to Bayes' law and used to predict the label. The OEM algorithm showed a better performance than BEM algorithm when class priors evolve along data sequences [26]. However, the prior estimation capability of OEM algorithm is insufficient. The reason is that few samples are available from the target distribution due to the prior evolution. Since sample size has great effects in parameter estimation, small-sample effects greatly degrade the performance of systems [27]. In concept drift, the prior evolution leads to the continuous change of distributions. Strictly speaking, only one sample is available from the target distribution, which makes the prior estimation in concept drift a difficult small sample estimation problem. In OEM algorithm, only the current sample is used to estimate the class priors, which leads to the poor performance of prior estimation.

In this paper, we propose a novel parameter estimation method called *transfer estimation*. It is inspired by transfer learning, a novel approach in machine learning community. Transfer learning aims to deal with small sample learning problems by transferring knowledge from related domain to the target domain [28]. Similar with transfer learning, transfer estimation makes uses of samples from not only the target distribution but also related distributions. Thus more information can be used to help improve the estimation performance. We analyze the property of this new estimation method. The theoretical condition is given which ensures transfer estimation a better performance. We apply transfer estimation method to OEM algorithm and get a new transductive algorithm. Experiments on both synthetic data sets and real-world data sets show that the new algorithm outperforms the OEM algorithm.

The rest of this paper is organized as follows. In Section 2, we introduce and analyze the proposed transfer estimation method. This new estimation method is applied to OEM algorithm and we get a new transductive algorithm. In Section 3, experimental results and analysis are reported. Finally, conclusions are given in Section 4.

2. Transfer estimation of class priors

2.1. Problem definition

We begin our discussion by introducing some notations used in this paper. Given an m-class classification problem, $\Omega=\{\omega_1,\ldots,\omega_m\}$ is the set of class labels. The data stream is an ordered set of samples $S=\{(x_1,y_1),\ldots,(x_t,y_t),\ldots\}$ as shown in Fig. 1. x_t is the attribute vector of the t th sample and $y_t \in \Omega$ is the class label, t can be from 1 to infinite. $P_t(x,\omega_i)=P(x|\omega_i)P_t(\omega_i)$ is the joint distribution with $\omega_i \in \Omega$, where $P(x|\omega_i)$ are the within-class probability densities and $P_t(\omega_i)$ are class priors. We assume the within-class probability densities remain unchanged and class priors evolve along data sequence. In the training sequence S_{tr}, y is known. And in the test sequence S_{ts}, y is unknown. The task is to predict the class label y_t of sample $(x_t,y_t) \in S_{ts}$ under the maximum a posteriori (MAP) rule:

$$\hat{y}_t = \arg\max_{\omega_i \in \Omega} P(\omega_i|x_t), \tag{1}$$

where $P(\omega_i|x_t)$ is the a posteriori probability of class ω_i of sample x_t, and $\hat{y}_t$ is the predicted label. Generative classifiers and discriminative classifiers can both give the a posteriori probability [29]. Discriminative classifiers directly compute the a posteriori probability. However, generative classifiers first estimate the within-class conditional probabilities and calculate the a posteriori probability using Bayes' Law as

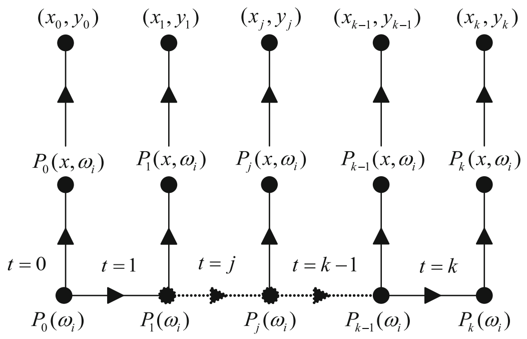

Fig. 1. Data stream with evolving class priors.

$$P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{\sum_{\omega_j \in \Omega} P(x|\omega_j)P_t(\omega_j)}. \tag{2}$$

It is commonly accepted that discriminative classifiers are better than generative classifiers [29]. In concept drift, however, generative classifiers are more suitable since the within-class distributions remain unchanged, which is experimentally validated in Ref. [26]. In this paper, we use generative classifiers as the basic classifiers. Naive Bayes classifier is a typical example of generative classifiers, which makes parametric assumption about the within-class conditional probabilities, for example a normal distribution assumption [30].

To calculate the a posterior probability shown by Eq. (2), the key issue is how to estimate the current class priors $P_t(\omega_i)$, which is a parameter estimation problem. x_t can be seen as drawn from the mixed distribution:

$$P_t(x) = \sum_{\omega_i \in \Omega} P(x|\omega_i)P_t(\omega_i), \tag{3}$$

where class priors $P_t(\omega_i)$ are the unknown parameters to estimate.

2.2. OEM algorithm

The state-of-art work of class prior estimation for static data is the BEM algorithm [23]. Yang et al. [26] proposed the OEM algorithm for evolving class prior estimation, which is an extension of BEM algorithm. The class prior estimation of OEM algorithm can be represented as follows:

$$\begin{cases} \hat{P}_0(\omega_i) = P_{ini}(\omega_i), \\ \hat{P}(\omega_i|x_k) = \dfrac{P(x_k|\omega_i)\hat{P}_{k-1}(\omega_i)}{\sum_{\omega_j \in \Omega} P(x_k|\omega_j)\hat{P}_{k-1}(\omega_j)}, \\ \hat{P}_k(\omega_i) = (1-\alpha)\hat{P}_{k-1}(\omega_i) + \alpha\hat{P}(\omega_i|x_k), \end{cases} \tag{4}$$

where $\hat{P}(\omega_i|x_k)$ is the estimated a posteriori probability of k th sample, $\hat{P}_k(\omega_i)$ is the updated class priors and α is the forgetting factor which is set empirically.

At the beginning $t=0$, class priors on the training set are initialized as $\hat{P}_0(\omega_i)=P_{ini}(\omega_i)$. And the initial classifier is trained on the independent training set with $\hat{P}_0(\omega_i)$. Based on EM algorithm, Yang et al. proved that the a posterior probability of class ω_i can be used as the estimator of class prior $P(\omega_i)$. At time $t=k-1$, we have the prior estimation $\hat{P}_{k-1}(\omega_i)$ of $P_{k-1}(\omega_i)$. At time $t=k$ with test sample (x_k,y_k), the a posterior probability estimation $\hat{P}(\omega_i|x)$ for generative classifier is calculated as in the second formula of Eqs. (4). Class priors at time $t=k$ are adjusted under the exponential forgetting rule as shown in the third formula of Eqs. (4).

Compared with BEM algorithm, OEM algorithm has shown its effectiveness on both synthetic and real data sets when class priors are non-stationary [26]. However, the prior estimation capability of OEM algorithm is insufficient. The estimation performance decreases considerably when the rate of the prior evolution increases. Forgetting factor α is an important parameter which controls the updating rate. The performance of OEM algorithm is very sensitive to the value of α. Since the value of α is set empirically, it is hard to apply OEM algorithm, especially to rapid evolution problems. That also results from the poor estimation capability of OEM algorithm. The essential reason is that the estimation of class priors in concept drift is a small sample parameter estimation problem.

2.3. Transfer estimation of class priors

Parameter estimation is important in statistics and information processing fields. $X=[x_1,\dots,x_n]$ are samples from an unknown distribution $P\in\mathbb{P}$ and θ is a real valued parameter related to P. Estimator $\tilde{\theta}=T(X)$ is a random variable and the expected value of $\tilde{\theta}$ is note as $g(\theta)=E[\tilde{\theta}]=E[T(X)]$. $\tilde{\theta}$ is called unbiased if $g(\theta)=\theta$.

Mean square error (MSE) $M^2(\tilde{\theta})$ is the common rule to evaluate the performance of an estimator $\tilde{\theta}$ in estimating θ. $M^2(\tilde{\theta})$ is defined:

$$M^2(\tilde{\theta})=E\{(\tilde{\theta}-\theta)^2\}=Var(\tilde{\theta})+b^2(\tilde{\theta}), \tag{5}$$

where $Var(\tilde{\theta})$ is the variance of $\tilde{\theta}$, $b^2(\tilde{\theta})$ is squared bias with $b(\tilde{\theta})=E(\tilde{\theta})-\theta$. Estimator $\tilde{\theta}_1$ is treated as a better one than $\tilde{\theta}_2$ if $M^2(\tilde{\theta}_1)\le M^2(\tilde{\theta}_2)$. According to the Fisher information inequality [31], $M^2(\tilde{\theta})$ has a lower bound called the Cramer–Rao lower bound:

$$M^2(\tilde{\theta})\ge\frac{[g'(\theta)]^2}{J(\theta)}. \tag{6}$$

If $\tilde{\theta}$ is unbiased, it turns out to be

$$M^2(\tilde{\theta})=Var(\tilde{\theta})\ge\frac{1}{J(\theta)}, \tag{7}$$

where $J(\theta)$ is the Fisher information. $J(\theta)$ is the information that X contains about the unknown parameter θ. The greater $J(\theta)$ is, the more information about θ is contained in X, the easier it is to distinguish θ from neighboring values. Therefore, we can estimate θ more accurately. In theory, the more samples we get from the identical distribution, the more information we have to help estimate the unknown parameters. If $x_1,\dots,x_n$ are sampled independently from the identical distribution, and $J(\theta)$ is the Fisher information about θ contained in a single sample, then the Fisher information about θ contained in $X=[x_1,\dots,x_n]$ is $nJ(\theta)$ [31]. In reality, however, we have limited samples to use when estimating the unknown parameters.

In concept drift problems, class priors evolve along the test sequence as shown in Fig. 1. Strictly speaking, $P_j(\omega_i)$ and $P_k(\omega_i)$ are not identical for any pair of j and k with $j\ne k$. Class priors $P_t(\omega_i)$ are the unknown parameters of distribution $P_t(x)=\sum_{\omega_i\in\Omega}P(x|\omega_i)P_t(\omega_i)$ as shown in Eq. (3). To estimate $P_t(\omega_i)$, only one sample x_t is available from distribution $P_t\,(x)$. As a result, the variance of estimator $\tilde{P}_t(\omega_i)$ is to be large, which makes MSE great. That is the origin of the poor performance of OEM algorithm on class prior estimation.

Small sample size problems are difficult in parameter estimation. Just the same problems occur in the fields of machine learning and pattern recognition. In knowledge engineering areas, significant success has been achieved by kinds of data mining and machine learning technologies. However, traditional methods work well based on the common assumption that the training and testing data are drawn from the same feature space and especially the same distribution. In many real-world applications, it is usually the case that no enough training samples are available in the target domain. Small-sample effects will easily contaminate the design and evaluation of a proposed system [27]. To deal with this problem, a novel approach called transfer learning stems from machine learning community recently. Different from the traditional methods, transfer learning allows the domains, tasks and distributions of samples used in training and testing data to be different. The most important point that transfer learning emphasizes is the transfer of knowledge across domains, tasks and distributions that are similar but not the same. It deals with small sample learning problems by transferring useful knowledge from related domain to the target domain [28]. Recently, kinds of transfer learning algorithms have been proposed [32–35]. It is proved that samples from different but related domain can really help solve the task in the target domain.

Inspired by the idea of transfer learning, we proposed a transfer estimation method to solve the small sample problems in class prior estimation of concept drift. The neighboring sample x_{t+k} of x_t is drawn from distribution $P_{t+k}(x)$. As shown in Fig. 1, $P_{t+k}(x)$ is different from $P_t(x)$ due to the class prior evolution. However, $P_{t+k}(x)$ and $P_t(x)$ are highly related with the same within-class distributions and share the same format of mixed distribution as shown in Eq. (3). We want to transfer the information contained in x_{t+k} to help estimate class priors $P_t(\omega_i)$. As in covariate shift learning problems [36], one straightforward way is to sum up the weighted estimator $\tilde{P}_{t+k}(\omega_i)$ of $P_{t+k}(\omega_i)$ and get a new estimator denoted by $\tilde{P}_{tr}(\omega_i)$ as follows:

$$\tilde{P}_{tr}(\omega_i)=\sum_{k=0}^{n-1}\alpha_k\tilde{P}_{t+k}(\omega_i)\quad\text{with}\ \sum_{k=0}^{n-1}\alpha_k=1, \tag{8}$$

where α_k are weighted parameters, and the numbers of neighboring samples used is $n-1$.

How to choose the value of weighted parameter α_k is situation-specific and should be considered according to the way of how class priors evolve. In practice however, we usually have little information on priors' evolving. Thus in this paper, we equally set the value of α_k as $1/n$, which is a general method. Then Eq. (8) turns out to be

$$\tilde{P}_{tr}(\omega_i)=\frac{1}{n}\sum_{k=0}^{n-1}\tilde{P}_{t+k}(\omega_i). \tag{9}$$

Different from estimators $\tilde{P}_{[t,\dots,t+n-1]}(\omega_i)$, estimator $\tilde{P}_{tr}(\omega_i)$ uses samples not only from the target distribution $P_t(x)$ but from distributions that are highly related with $P_t(x)$. We call this estimation method as *transfer estimation*, and $\tilde{P}_{tr}(\omega_i)$ as *transfer estimator* in this paper. Using samples of related distributions may increase the squared bias of the estimator, but will effectively decrease the estimation variance with more information used. Estimation performance will be improved as long as the variance is reduced in a larger magnitude. In the following, we will prove that transfer estimation can usually give a better estimation.

Now we give a discussion on the property of transfer estimator. First, we introduce some notations and definitions used in the following theorems.

Notation. θ_k is the real valued parameter of distribution $P(\theta_k)$, $k=1,\dots,n$. $\theta_{[1,\dots,n]}$ is a shorthand for $\theta_1,\dots,\theta_n$. The mean of $\theta_{[1,\dots,n]}$ is noted as $\overline{\theta}=1/n\sum_{k=1}^{n}\theta_k$. Estimator of parameter θ_k is noted as $\tilde{\theta}_k$. Transfer estimator is noted as $\tilde{\theta}_{tr}$, with $\tilde{\theta}_{tr}=(1/n)\sum_{k=1}^{n}\tilde{\theta}_k$.

Definition. $\tilde{\theta}_{tr}$ is a *better estimator* as a whole than $\tilde{\theta}_{[1,\ldots,n]}$, provided the following condition holds

$$\sum_{k=1}^{n} E(\tilde{\theta}_{tr}-\theta_k)^2 \le \sum_{k=1}^{n} E(\tilde{\theta}_k-\theta_k)^2. \tag{10}$$

The definition above is based on mean square error rule, where $\sum_{k=1}^{n} E(\tilde{\theta}_{tr}-\theta_k)^2$ and $\sum_{k=1}^{n} E(\tilde{\theta}_k-\theta_k)^2$ are the overall estimation performance of $\tilde{\theta}_{tr}$ and $\tilde{\theta}_{[1,\ldots,n]}$, respectively. It defines one way of evaluating transfer estimator. Based on the notation and definition, the property of the transfer estimator is included in Theorems 1 and 2 as below.

Theorem 1. *Assume estimators $\tilde{\theta}_{[1,\ldots,n]}$ are unbiased. Then $\tilde{\theta}_{tr}$ is a* better estimator *if and only if the following condition holds*

$$\sum_{k=1}^{n} (\theta_k^2-\overline{\theta}^2) \le \frac{n-1}{n}\sum_{k=1}^{n} Var(\tilde{\theta}_k). \tag{11}$$

where $Var(\tilde{\theta}_k)$ are the variance of $\tilde{\theta}_k$, $k=1,\ldots,n$.

When estimators $\tilde{\theta}_{[1,\ldots,n]}$ are biased, we need consider the influence of estimation bias, having Theorems 2 as below.

Theorem 2. *$\tilde{\theta}_{tr}$ is a* better estimator *provided the following condition holds*

$$\sum_{k=1}^{n} (\theta_k^2-\overline{\theta}^2) + \frac{2}{n}\sum_{k=1}^{n} \{b(\tilde{\theta}_k)(\overline{\theta}-\theta_k)\} \le \frac{n-1}{n}\sum_{k=1}^{n} Var(\tilde{\theta}_k), \tag{12}$$

where $b(\tilde{\theta}_k)$ and $Var(\tilde{\theta}_k)$ are the bias and variance of $\tilde{\theta}_k$ respectively, $k=1,\ldots,n$.

The proof of both theorems is given in the appendix of this paper. Theorems 1 and 2 discuss the property of transfer estimator under different situations. In Theorem 1, Eq. (11) gives the necessary and sufficient condition that ensures transfer estimator $\tilde{\theta}_{tr}$ a better one when estimators $\tilde{\theta}_{[1,\ldots,n]}$ are unbiased. $\sum_{k=1}^{n}(\theta_k^2-\overline{\theta}^2)$ characterizes the difference among distribution parameters $\theta_{[1,\ldots,n]}$. It measures the increased squared bias introduced by transfer estimator. How to define the evolution rate in concept drift is important and there is no definite method [13]. $\sum_{k=1}^{n}(\theta_k^2-\overline{\theta}^2)$ can be seen as the influence of the parameter evolution on the estimation performance of $\tilde{\theta}_{tr}$. Therefore, the average difference $\sum_{k=1}^{n}(\theta_k^2-\overline{\theta}^2)/n$ can be used to quantitatively characterize the evolution rate, where n is the number of samples that estimator $\tilde{\theta}_{tr}$ uses. In Eq. (11), $\sum_{k=1}^{n} Var(\tilde{\theta}_k)$ is the sum of variance. It is the overall performance of unbiased estimators $\tilde{\theta}_{[1,\ldots,n]}$. And $(n-1)/n\sum_{k=1}^{n} Var(\tilde{\theta}_k)$ quantifies the reduced variance by transfer estimator. Theorem 2 supplies one sufficient condition when estimators $\tilde{\theta}_{[1,\ldots,n]}$ are biased. Compared with Eq. (11), Eq. (12) has an extra term $2/n\sum_{k=1}^{n}\{b(\tilde{\theta}_k)(\overline{\theta}-\theta_k)\}$. It reflects the influence of estimation bias combined with parameter evolution. However, this influence is weak when n is relatively large. In conclusion, the superiority of transfer estimator is mainly decided on the comparison between $\sum_{k=1}^{n}(\theta_k^2-\overline{\theta}^2)$ (i.e., parameter evolution influence) and $(n-1)/n\sum_{k=1}^{n} Var(\tilde{\theta}_k)$ (i.e., estimation variance influence). Transfer estimator will be a good choice when the condition of $\sum_{k=1}^{n}(\theta_k^2-\overline{\theta}^2) \le (n-1)/n\sum_{k=1}^{n} Var(\tilde{\theta}_k)$ is satisfied. In most cases, especially for the small-sample ones, this condition can be satisfied because the left term is rather limited due to the continuous change of parameters, and at the same time, the right term will be much larger when n is large (e.g., 30–100). We will also exemplify this through experiments in Section 3.

2.4. Tr-OEM algorithm

The estimation of class priors in concept drift is a typical small sample estimation problem. Here we apply the transfer estimation method to OEM algorithm, and get a new algorithm called transfer OEM algorithm, which is denoted by Tr-OEM. Tr-OEM algorithm is succinctly described as

$$\begin{cases} \hat{P}_0(\omega_i) = P_{ini}(\omega_i), \\ \hat{P}(\omega_i|x_k) = \dfrac{P(x_k|\omega_i)\hat{P}_{k-1}(\omega_i)}{\sum_{\omega_j \in \Omega} P(x_k|\omega_j)\hat{P}_{k-1}(\omega_j)}, \\ \hat{P}_{tr}(\omega_i) = \dfrac{1}{n}\sum_{l=0}^{n-1}\hat{P}(\omega_i|x_{k+l}), \\ \hat{P}_k(\omega_i) = (1-\alpha)\hat{P}_{k-1}(\omega_i) + \alpha\hat{P}_{tr}(\omega_i), \end{cases} \tag{13}$$

where $\hat{P}_{tr}(\omega_i)$ is the transfer estimator, n is the number of samples that $\hat{P}_{tr}(\omega_i)$ uses.

3. Experiments

To demonstrate the validity of transfer estimation in concept drift, experiments on both synthetic and real data sets were conducted. All the data sets were formerly used in Ref. [26]. In this section, we first introduced the evaluation criteria as well as the data sets. Subsequently, we tested the performance of OEM algorithm [26] and the proposed Tr-OEM algorithm on these data sets. Performance on both prior estimation and classification was evaluated. In all these experiments, we used Naive Bayes classifier to calculate the a posteriori probability.

3.1. Evaluation criteria

Average variational distance (*AVD*): We know the true class priors of the synthetic data sequences. In the experiments on synthetic data sets, we will evaluate the prior estimation performance of algorithms by measuring the divergence between the estimated prior probability $\hat{P}_t(\omega_i)$ and the true prior probability $P_t(\omega_i)$. In probability theory and statistics, several divergence measures between two probability distributions have been introduced [37]. And the variational distance is a straightforward measure [38]. Variation distance between $\hat{P}_t(\omega_i)$ and $P_t(\omega_i)$ is calculated as follows:

$$V(\hat{P}_t, P_t; \Omega) = \sum_{\omega_i \in \Omega} |\hat{P}_t(\omega_i) - P_t(\omega_i)|, \tag{14}$$

where t is the index of test samples, $\hat{P}_t(\omega_i)$ is the estimated prior probability and $P_t(\omega_i)$ is the true prior probability. Based on variational distance, we define the average variational distance $AVD(k)$ as follows:

$$AVD(k) = \frac{\sum_{t=1}^{k} V(\hat{P}_t, P_t; \Omega)}{k}. \tag{15}$$

Accumulated accuracy (*AACC*): In this paper, classification accuracy (*ACC*) is used as the evaluation criteria of classification performance as in Ref. [23]. We use *ACC* to compare algorithms' overall performance on the whole data sets. Assume n samples are contained in the data set, of which m samples are correctly classified. The classification accuracy on this data set will be defined as $ACC = m/n$. To reflect the evolution of the classification accuracy along the data sequence, we need to dynamically evaluate the classification performance of algorithms. Therefore we define the accumulated accuracy (*AACC*), where $AACC(k)$ is the *ACC* of the first k samples of the test sequence. Assume m_k of k samples are correctly classified, then we have $AACC(k) = m_k/k$.

3.2. Synthetic data

Synthetic data set "1D2CV" is a two-class univariate data sequence with noticeable concept drift [26]. In 1D2CV data set, two classes were independently drawn from two Gaussian distributions $N(0,1)$ and $N(1,1.5)$. 500 samples were generated for each class. Samples in the test sequence were drawn according to specified varying class distribution. The generation of the class priors is as

$$\tilde{P}_t(\omega_i) = \left| i + \frac{5\times i\times t}{L} + 10\times \sin\left(\frac{4\times i\times t\times c}{L}\right)\right| \tag{16}$$

and

$$P_t(\omega_i) = \frac{\tilde{P}_t(\omega_i)}{\sum_{\omega_j \in \Omega} \tilde{P}_t(\omega_j)}, \tag{17}$$

where t is the index of the sample, $i \in [1,2]$, and c is a tuning parameter. The length of the test sequence is L=10,000. c controls the evolution rate of class priors. The class priors evolve more rapidly when c increases, as depicted in Fig. 2.

3.3. Real data

To evaluate the performance of algorithms on the real-world data sets, we performed experiments on a large scale real-world data sequence called "Forest Covertype" as in Ref. [39]. The Forest Covertype data set is a multi-class problem with varying class priors. The whole data set contains totally 581,012 samples belonging to seven classes. Each sample consists of 54 attributes, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables. The first 15,120 samples were used as training samples and the last 565,892 records were used to construct the test sequences. In real data sets, we do not know the true class priors of data sequences. To estimate the priors of one sample in the real data set, we use the true labels of its neighbors and calculate the frequencies, which are treated as the approximate values of the target class priors. Different numbers of neighbors can be used to calculate the frequencies. 2000 neighbors were used in this paper. The Forest Covertype data set is a seven-class problem. To save space here, Fig. 3 just shows the approximate class priors of Classes 1 and 2. We can see significant concept drift on Forest Covertype data set, e.g, Class 1 is prominent (80%) at the beginning of the sequence and its proposition decreases below 30% at the middle of the sequence.

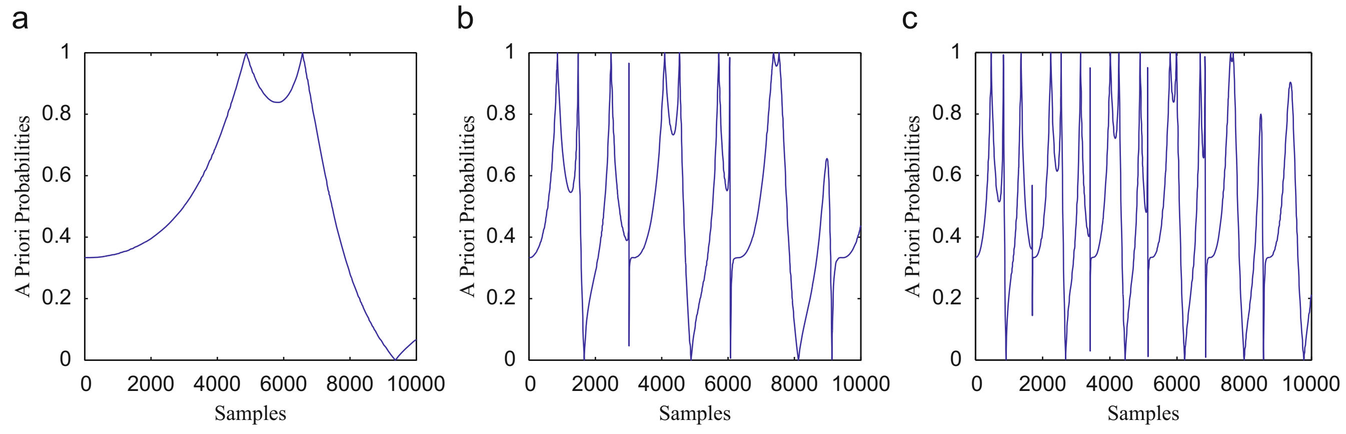

Fig. 2. Evolutions of $P_t(\omega_1)$ on 1D2CV data sets with different tune parameters c: (a) c=1; (b) c=5; and (c) c=9.

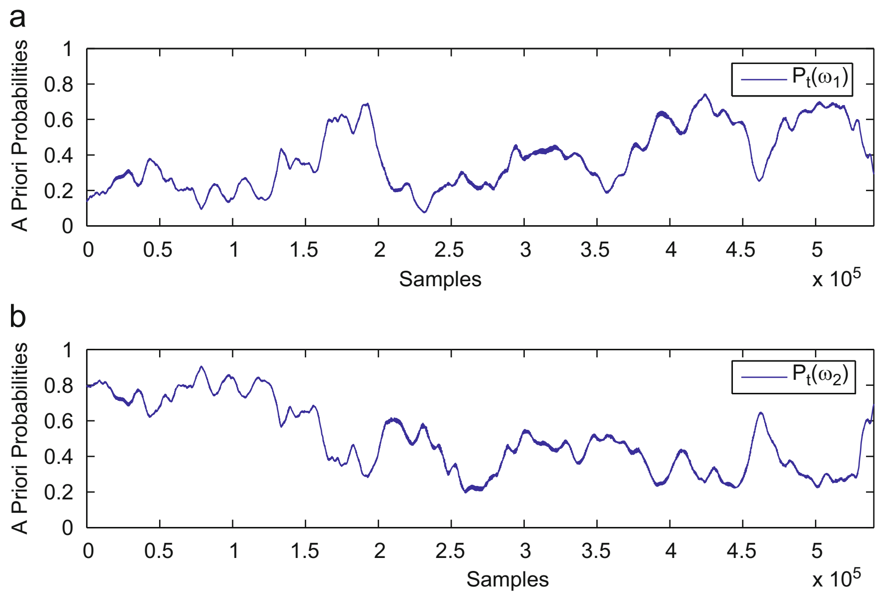

Fig. 3. Evolutions of class priors on the Forest Covertype data set: (a) $P_t(\omega_1)$ and (b) $P_t(\omega_2)$.

3.4. Results on synthetic data

In synthetic data, we have the true class priors $P(\omega_i)$ and calculate the average variational distance (*AVD*) between $P(\omega)$ and the estimated priors $\hat{P}(\omega_i)$ by algorithms. Table 1 lists the *AVD* of OEM and Tr-OEM algorithms using different forgetting factors α on the 1D2CV data sets with different tuning parameters *c*. It is shown that Tr-OEM algorithm can improve the performance on class prior estimation under different forgetting factors and tuning parameters, consistently and considerably. The performance of both OEM and Tr-OEM algorithms decreases as prior evolution rate increases. However, the performance of Tr-OEM algorithm is relatively stable. And the worse OEM algorithm performs, the larger the improvement by Tr-OEM algorithm will be, which follows the conclusion of theorems. In practice, forgetting factor α is set empirically. The performance of OEM is sensitive to α. When $\alpha \geq 0.5$, OEM algorithm is unacceptable and results are not shown. However, Tr-OEM algorithm is relatively robust to α.

To have a more straightforward view, the estimated values of class priors $\hat{P}_t(\omega_1)$ by OEM and Tr-OEM algorithms are shown along the data sequences in Fig. 4. The dash-dot lines are the true class priors $P_t(\omega_1)$ on 1D2CV data sets with $c=1$, 5, and 9, respectively. The solid lines are the estimated priors $\hat{P}_t(\omega_1)$. It is shown clearly that Tr-OEM algorithm gives more accurate and smooth curves of $\hat{P}_t(\omega_1)$ compared with the curves by OEM algorithm.

In the last paragraph of Section 2.3,we discussed the property of transfer estimator. According to the theoretic analysis,there are mainly two factors that decide the superiority of transfer estimator. $\sum_{k=1}^{n}(\theta_k^2-\overline{\theta}^2)$, noted as Evo_{infl},reflects the influence of prior evolution. $(n-1)/n\sum_{k=1}^{n} Var(\tilde{\theta}_k)$, noted as Var_{infl},reflects the influence of estimation variance of OEM algorithm. We calculated the values of Evo_{infl} and Var_{infl} under all situations in Table 1. It was shown that the condition of Eq. (11) is consistently satisfied. Due to space limit,we only list the results of partial situations in Table 2. To calculate Var_{infl},we ran OEM algorithm 30 times under every situation,and used the 30 estimation values to calculate the mean value and variance of every estimator along the sequence. It is presented in Table 2 that Var_{infl} is always larger than Evo_{infl},which satisfies the condition of Eq. (11). It explains why transfer estimator gives a better estimation in Table 1. Table 2 also lists the value of $2/n\sum_{k=1}^{n}\{b(\tilde{\theta}_k)(\overline{\theta}-\theta_k)\}$ in Eq. (12), noted as $Bias_{infl}$. It reflects the influence of estimation bias combined with prior evolution. The condition of Eq. (12) is still satisfied when considering $Bias_{infl}$. As presented in Table 2, $Bias_{infl}$ is very small. Therefore, the influence of $Bias_{infl}$ is very weak. It illustrates that the superiority of transfer estimator is mainly decided by Evo_{infl} and Var_{infl}.

Table 3 lists the classification accuracy (*ACC*) of OEM and Tr-OEM algorithms using different forgetting factors α on 1D2CV data sets with different tuning parameters *c*. It is shown that Tr-OEM algorithm also improved the *ACC* consistently under different forgetting factors and tuning parameters, similar with the situation in class prior estimation. As a whole, the worse OEM algorithm performs, the more superior Tr-OEM algorithm is. Transfer estimation also makes the performance of Tr-OEM algorithm less sensitive to α than OEM algorithm, which is very important in practical applications.

In concept drift, the ideal estimation algorithm is that the estimated class priors are exactly equal to the true class priors, that is $\hat{P}_t(\omega_i)=P_t(\omega_i)$. It is impossible in reality, however, because we even have no access to the true class priors of real data. But in synthetic data, we can get the true values of class priors and design the ideal algorithm that adjusts classifiers according to the true class priors. The classification accuracy of this ideal algorithm is noted as ACC_{Ideal}. ACC_{Ideal} is the upper bound of accuracy of algorithms for concept drift. ACC_{Ideal} on the 1D2CV data sets with different tuning parameters are presented in Table 4. To have a better understanding on the performance of OEM and Tr-OEM algorithms, we compare the classification accuracy with ACC_{Ideal}. As presented in Table 3, forgetting factor α has significant effect on classification performance. We only compare the best *ACC* of OEM algorithm and Tr-OEM algorithm under different tune parameters *c* with ACC_{Ideal} as in Table 4. It shows that Tr-OEM algorithm exceeds OEM algorithm, especially when OEM performs badly. As the evolution rate increased, the superiority of Tr-OEM to OEM was increased. And the best *ACC* of Tr-OEM algorithm is much nearer to ACC_{Ideal} than that of OEM algorithm.

Table 1
Comparison of $AVD(k)$ between OEM [26] and Tr-OEM algorithms on 1D2CV data sets ($k=10{,}000$, $n=50$, $P_{ini}(\omega_1)=0.5$).

	AVD (%)	$\alpha=0.05$	$\alpha=0.1$	$\alpha=0.15$	$\alpha=0.2$	$\alpha=0.25$	$\alpha=0.3$	$\alpha=0.35$	$\alpha=0.4$	$\alpha=0.45$	$\alpha=0.5$
$c=1$	OEM	**14.88**	17.96	21.7	25.31	28.81	31.8	35.73	39.02	41.94	46.55
	Tr-OEM	**13.85**	15.23	16.71	17.83	18.64	19.24	19.71	20.09	20.39	20.64
$c=5$	OEM	24.22	**22.75**	24.89	28.04	31.56	35.37	39.32	43.47	47.71	52.17
	Tr-OEM	20.2	**17.07**	17.14	17.86	18.6	19.28	19.88	20.4	20.83	21.19
$c=9$	OEM	30.84	**28.11**	29.09	31.57	34.72	38.07	41.43	44.98	48.67	52.27
	Tr-OEM	25.32	20.44	**19.78**	20.26	20.97	21.65	22.25	22.77	23.21	23.6
$c=13$	OEM	30.93	**28.35**	28.82	30.47	32.75	35.45	38.51	41.91	45.54	49.02
	Tr-OEM	25.08	20.12	**19.28**	19.64	20.3	20.97	21.6	22.16	22.63	23.05
$c=17$	OEM	32.38	28.22	**27.75**	28.96	30.99	33.48	36.29	39.26	42.44	45.9
	Tr-OEM	25.67	19.89	**19.04**	19.48	20.34	21.24	22.07	22.79	23.4	23.93
$c=21$	OEM	35.37	32.92	**32.62**	33.66	35.57	38.06	41.02	44.33	47.73	51.05
	Tr-OEM	27.98	21.95	20.18	**20.16**	20.63	21.24	21.86	22.46	23.02	23.52
$c=25$	OEM	37.32	35.27	**34.7**	35.5	35.17	39.43	42.11	45.1	48.36	51.73
	Tr-OEM	30.9	25.1	25.84	25.79	**23.43**	24.29	25.15	25.93	26.62	27.24

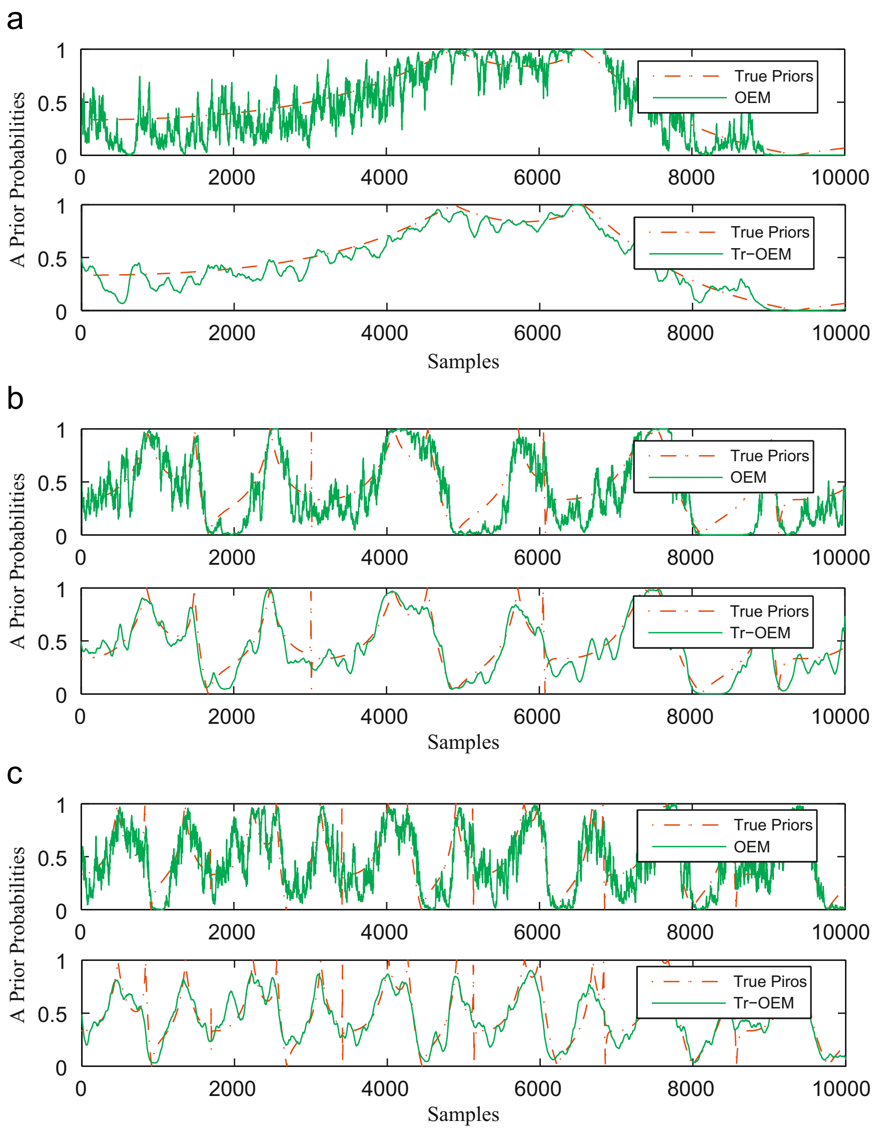

Fig. 4. The true values and estimated values of $P_t(\omega_1)$ on 1D2CV data sets ($P_{ini}(\omega_1)=0.5$, $n=50$, $\alpha=0.2$). (a) $c=1$; (b) $c=5$ and (c) $c=9$.

Table 2
Comparison of Var_{infl}, Evo_{infl} and $Bias_{infl}$ of OEM algorithm [26] on 1D2CV data sets ($\alpha=0.2$, $k=10{,}000$, $n=50$, $P_{ini}(\omega_1)=0.5$).

	$c=1$	$c=5$	$c=9$	$c=13$	$c=17$	$c=21$	$c=25$
Var_{infl}	173.6	206.4	199.1	213.17	216.4	218.0	217.7
Evo_{infl}	0.1	9.8	20.6	34.1	55.8	76.3	95.3
$Bias_{infl}$	0.0008	0.19	0.45	0.76	1.21	1.65	2.26

3.5. Results on real data

As in Ref. [26], we performed experiments on the large real-world data "Forest Covertype". As described above, the first 15,120 samples were used as the training samples. The class frequencies on those training samples are equally 0.1429, which were used as the initial class priors for OEM and Tr-OEM algorithms. Table 5 lists the classification accuracy of OEM and Tr-OEM algorithms on Forest Covertype data set under different forgetting factors α. It is shown that the performance of OEM algorithm is very sensitive to the value of α. And α can only vary in a very narrow interval ($\alpha <= 0.05$) to get a reasonable performance. The performance of OEM algorithm deteriorates considerably when $\alpha \geq 0.1$. The value of α is set empirically, therefore it is inconvenient to apply OEM algorithm in practice. Compared with OEM algorithm, Tr-OEM algorithm gives a better performance in most cases ($\alpha \geq 0.05$). And the performance is relative robust to α, which makes it more convenient to apply Tr-OEM algorithm in real-world applications. The best *ACC* of OEM algorithm presented in Table 5 is **63.54**% with $\alpha=0.01$. And the best *ACC* of Tr-OEM algorithm we found is **67.54**% with $\alpha=0.2$. To dynamically evaluate the classification performance, we compare the *AACC* curves of OEM algorithm ($\alpha=0.01$) and Tr-OEM algorithm($\alpha=0.2$) in Fig. 5. As shown in Fig. 5, Tr-OEM algorithm always performs better than OEM algorithm along the data sequence.

The evolution rate of class priors is an important factor in the experiments on synthetic data. The more rapidly class priors evolve, the harder it will be to estimate the evolving priors. We demonstrated the validity of Tr-OEM algorithm by performing experiments on real data sets with higher evolution rates. Data sets were generated by drawing samples from Forest Covertype as the following method. Totally $N=565{,}892$ samples are available in the test sequence of Forest Covertype. First, we divide the 565,892 test samples equally into $[N/s]$ subgroups following the

Table 3
Comparison of *ACC* between OEM [26] and Tr-OEM algorithms on 1D2CV data sets ($n=50$, $P_{ini}(\omega_1)=0.5$).

	ACC (%)	$\alpha=0.05$	$\alpha=0.1$	$\alpha=0.15$	$\alpha=0.2$	$\alpha=0.25$	$\alpha=0.3$	$\alpha=0.35$	$\alpha=0.4$	$\alpha=0.45$	$\alpha=0.5$
$c=1$	OEM	**78.82**	78.09	77.34	76.88	76.28	75.72	74.93	73.99	72.98	71.44
	Tr-OEM	**79.03**	78.85	78.8	78.51	78.37	78.35	78.22	78.12	78	78.03
$c=5$	OEM	74.03	**74.55**	74.48	73.9	73.42	72.7	71.91	71.01	69.68	68.28
	Tr-OEM	75	75.75	**75.92**	75.88	75.67	75.51	75.48	75.36	75.37	75.28
$c=9$	OEM	72.54	**73.66**	73.53	73.04	72.67	72.15	71.28	70.51	69.39	68.46
	Tr-OEM	74.38	75.88	**76.05**	75.91	75.85	75.79	75.67	75.83	75.75	75.54
$c=13$	OEM	71.72	73.37	**73.79**	73.54	73.25	72.61	72.09	71.54	70.55	69.48
	Tr-OEM	73.35	75.17	**75.58**	75.3	75	75.13	74.77	74.67	74.65	74.68
$c=17$	OEM	71.48	72.78	**73.23**	73.05	72.72	72.3	71.82	71.37	70.77	69.97
	Tr-OEM	73.25	74.9	75.66	**75.77**	75.68	75.48	75.4	75.24	75.4	75.1
$c=21$	OEM	69.35	70.24	71.02	**71.31**	70.91	70.27	69.71	69.03	68.53	67.53
	Tr-OEM	71.53	73.61	73.93	74.3	**74.35**	74.23	74.18	74.16	74.06	74
$c=25$	OEM	70.07	71.18	71.6	**71.6**	71.32	71.01	70.29	69.64	68.71	67.56
	Tr-OEM	72.75	74.33	75.03	**75.18**	75.02	74.69	74.34	74.07	73.92	73.81

Table 4
Comparison of the best *ACC* of OEM [26] and Tr-OEM algorithms with ACC_{Ideal} on 1D2CV data sets.

ACC (%)	$c=1$	$c=5$	$c=9$	$c=13$	$c=17$	$c=21$	$c=25$
OEM	78.82	74.55	73.66	73.79	73.23	71.31	71.6
Tr-OEM	79.03	75.92	76.05	75.58	75.77	74.35	75.18
ACC_{Ideal}	80.29	76.9	77.37	76.99	77.25	77.08	77.21

Table 5
ACC of OEM [26] and Tr-OEM algorithms on Forest Covertype data set.

α	0.01	0.05	0.1	0.2	0.3	0.4	0.5
OEM (%)	**63.54**	62.38	60.86	59.29	58.92	58.67	58.41
Tr-OEM (%)	62.94	64.15	65.19	**67.54**	66.37	64.76	64.25

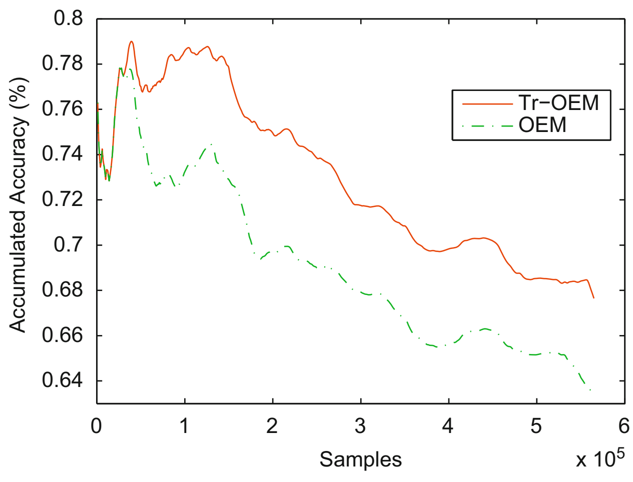

Fig. 5. *AACC* of OEM [26] and Tr-OEM algorithms on Forest Covertype data set.

Table 6
ACC of OEM [26] and Tr-OEM algorithms on FC-3 data set.

α	0.01	0.05	0.1	0.2	0.3	0.4	0.5
OEM (%)	**61.90**	57.68	54.52	53.72	53.11	52.61	51.97
Tr-OEM (%)	63.69	64.12	65.78	67.03	**67.32**	67.10	65.74

Table 7
ACC of OEM [26] and Tr-OEM algorithms on FC-5 data set.

α	0.01	0.05	0.1	0.2	0.3	0.4	0.5
OEM (%)	55.87	56.26	**57.31**	57.22	56.16	55.16	54.40
Tr-OEM (%)	62.12	63.18	64.89	**66.87**	66.52	66.27	66.11

order of the sequence, where s is the number of samples in every subgroup. One sample will be drawn randomly with equal probability from every subgroup. $[N/s]$ samples are drawn in total, which constitute a new real data set with a prior evolution rate higher than that of Forest Covertype. Parameter s controls the evolution rate of the newly generated data sets. Within a certain range of s, evolution rate increases as s increases. In this paper, we set the value of s as 3 and 5. Two data sets were generated, noted as FC-3 and FC-5, respectively, for convenience.

Experiments were performed on FC-3 and FC-5 data sets. Tables 6 and 7 list the results, respectively. It is shown that Tr-OEM algorithm exceeds OEM algorithm consistently under different values of α on both data sets. And the worse OEM algorithm performs, the larger the improvement by Tr-OEM algorithm will be, which is just the same situation as that on the synthetic data. By comparing the results in Tables 5–7, we can see that the performance of OEM algorithm deteriorates significantly as the prior evolution rate increases. The best *ACC* of OEM algorithm drops from **63.54**% to **57.31**%. However, the influence of evolution rates on the performance of Tr-OEM algorithm is very weak. The best *ACC* of Tr-OEM algorithm are **67.54**%, **67.32**% and **66.87**%, respectively, as the evolution rate increases.

4. Conclusions

In this paper, we proposed a novel parameter estimation method called *transfer estimation*. Similar with transfer learning algorithms in machine learning, transfer estimation makes use of samples not only from the target distribution but also from related distributions. We analyzed the property of transfer estimator theoretically and gave the condition that ensures it a better performance. Transfer estimation is of great help in parameter estimation when samples from the target distribution are scarce. We applied this new method to the class prior estimation of data streams with concept drift. And based on the OEM algorithm, we got a new algorithm called Tr-OEM. Experiments on both synthetic and real data sets showed that Tr-OEM algorithm exceeds OEM algorithm on prior estimation and classification. As the prior evolution rate increases, the performance of OEM algorithm deteriorates significantly. However, the influence of evolution rate on Tr-OEM algorithm is much weaker. OEM algorithm is sensitive to parameters' value, while Tr-OEM algorithm is relatively robust. Therefore, it is more convenient to use Tr-OEM algorithm for practical applications.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Nos. 60721003, 60673106 and 60573062) and the Specialized Research Fund for the Doctoral Program of Higher Education.

Appendix A. Proof of theorem 2

We first give the proof of Theorem 2. Some results will be used in the proof of Theorem 1.

Proof. The difference between θ_i and θ_j is noted as $\delta_{ij}=\theta_i-\theta_j$. Then the estimation performance of $\tilde{\theta}_i$ on θ_j is

$$E[(\tilde{\theta}_i-\theta_j)^2]=E(\tilde{\theta}_i-\theta_i+\delta_{ij})^2=E(\tilde{\theta}_i-\theta_i)^2+2\delta_{ij}b(\tilde{\theta}_i)+\delta_{ij}^2. \tag{18}$$

The estimation performance of transfer estimator $\tilde{\theta}_{tr}$ on θ_k is

$$E[(\tilde{\theta}_{tr}-\theta_k)^2]=E\left[\left(\frac{1}{n}\sum_{i=1}^{n}\tilde{\theta}_i-\theta_k\right)^2\right]=\frac{1}{n^2}\sum_{i=1}^{n}E(\tilde{\theta}_i-\theta_k)^2+\frac{2}{n^2}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}E(\tilde{\theta}_i-\theta_k)E(\tilde{\theta}_j-\theta_k). \tag{19}$$

Substituting (18) into (19), we have

$$E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n^2}\sum_{i=1}^{n}\{E(\tilde{\theta}_i-\theta_i)^2+2\delta_{ik}b(\tilde{\theta}_i)+\delta_{ik}^2\}+\frac{2}{n^2}\sum_{i=1}^{n}(b(\tilde{\theta}_i)+\delta_{ik})\sum_{j=1,j\neq i}^{n}(b(\tilde{\theta}_j)+\delta_{jk}), \tag{20}$$

that is

$$E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n^2}\sum_{i=1}^{n}\{Var(\tilde{\theta}_i)^2+b^2(\tilde{\theta}_i)+2\delta_{ik}b(\tilde{\theta}_i)+\delta_{ik}^2\}+\frac{2}{n^2}\sum_{i=1}^{n}\delta_{ik}\sum_{j=1,j\neq i}^{n}\delta_{jk}+\frac{2}{n^2}\sum_{i=1}^{n}b(\tilde{\theta}_i)\sum_{j=1,j\neq i}^{n}b(\tilde{\theta}_j)+\frac{2}{n^2}\sum_{i=1}^{n}b(\tilde{\theta}_i)\sum_{j=1,j\neq i}^{n}\delta_{jk}$$
$$+\frac{2}{n^2}\sum_{i=1}^{n}\delta_{ik}\sum_{j=1,j\neq i}^{n}b(\tilde{\theta}_j). \tag{21}$$

Note that

$$\frac{1}{n^2}\sum_{i=1}^{n}b^2(\tilde{\theta}_i)+\frac{2}{n^2}\sum_{i=1}^{n}b(\tilde{\theta}_i)\sum_{j=1,j\neq i}^{n}b(\tilde{\theta}_j)=\frac{1}{n^2}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2, \tag{22}$$

$$\sum_{i=1}^{n}b(\tilde{\theta}_i)\sum_{j=1,j\neq i}^{n}\delta_{jk}=\sum_{i=1}^{n}\delta_{ik}\sum_{j=1,j\neq i}^{n}b(\tilde{\theta}_j), \tag{23}$$

$$\sum_{i=1}^{n}b(\tilde{\theta}_i)\sum_{j=1,j\neq i}^{n}\delta_{jk}+\sum_{i=1}^{n}\delta_{ik}b(\tilde{\theta}_i)=\sum_{i=1}^{n}b(\tilde{\theta}_i)\sum_{j=1}^{n}\delta_{jk}. \tag{24}$$

By (21)–(24), the estimation performance of $\tilde{\theta}_{tr}$ on θ_k will be

$$E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n^2}\sum_{i=1}^{n}Var(\tilde{\theta}_i)^2+\frac{1}{n^2}\left\{\sum_{i=1}^{n}\delta_{ik}\right\}^2+\frac{1}{n^2}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2+\frac{4}{n^2}\sum_{i=1}^{n}\delta_{ik}\sum_{j=1}^{n}b(\tilde{\theta}_j)-\frac{2}{n^2}\sum_{i=1}^{n}\delta_{ik}b(\tilde{\theta}_i). \tag{25}$$

Now we consider the overall performance of $\tilde{\theta}_{tr}$ on $\theta_{[1,\ldots,n]}$, that is $\sum_{k=1}^{n}E(\tilde{\theta}_{tr}-\theta_k)^2$. By (25), we have

$$\sum_{k=1}^{n}E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)^2+\frac{1}{n^2}\sum_{k=1}^{n}\left\{\sum_{i=1}^{n}\delta_{ik}\right\}^2+\frac{1}{n}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2+\frac{4}{n^2}\sum_{k=1}^{n}\sum_{i=1}^{n}\delta_{ik}\sum_{j=1}^{n}b(\tilde{\theta}_j)-\frac{2}{n^2}\sum_{k=1}^{n}\sum_{i=1}^{n}\delta_{ik}b(\tilde{\theta}_i). \tag{26}$$

Note that

$$\frac{1}{n^2}\sum_{k=1}^{n}\left\{\sum_{i=1}^{n}\delta_{ik}\right\}^2 = \sum_{j=1}^{n}(\theta_j^2-\overline{\theta}^2), \tag{27}$$

$$\sum_{k=1}^{n}\sum_{i=1}^{n}\delta_{ik}=0. \tag{28}$$

By (27)–(28), (26) can be simplified as

$$\sum_{k=1}^{n}E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)^2+\sum_{j=1}^{n}(\theta_j^2-\overline{\theta}^2)+\frac{1}{n}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2+\frac{2}{n^2}\sum_{i=1}^{n}\left\{b(\tilde{\theta}_i)\sum_{k=1}^{n}\delta_{ki}\right\}, \tag{29}$$

that is

$$\sum_{k=1}^{n}E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)^2+\sum_{j=1}^{n}(\theta_j^2-\overline{\theta}^2)+\frac{1}{n}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2+\frac{2}{n}\sum_{i=1}^{n}\{b(\tilde{\theta}_i)(\overline{\theta}-\theta_i)\}. \tag{30}$$

To make $\tilde{\theta}_{tr}$ better than $\tilde{\theta}_{[1,\dots,n]}$, (10) should be satisfied, that is

$$\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)+\sum_{j=1}^{n}(\theta_j^2-\overline{\theta}^2)+\frac{1}{n}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2+\frac{2}{n}\sum_{i=1}^{n}\{b(\tilde{\theta}_i)(\overline{\theta}-\theta_i)\}\le\sum_{k=1}^{n}Var(\tilde{\theta}_k)+\sum_{k=1}^{n}b^2(\tilde{\theta}_k). \tag{31}$$

According to *Cauchy–Schwarz* inequality, we have

$$\frac{1}{n}\left\{\sum_{i=1}^{n}b(\tilde{\theta}_i)\right\}^2\le\sum_{k=1}^{n}b^2(\tilde{\theta}_k). \tag{32}$$

By (31) and (32), we get

$$\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)+\sum_{i=1}^{n}(\theta_i^2-\overline{\theta}^2)+\frac{2}{n}\sum_{i=1}^{n}\{b(\tilde{\theta}_i)(\overline{\theta}-\theta_i)\}\le\sum_{i=1}^{n}Var(\tilde{\theta}_i). \tag{33}$$

Transfer estimator $\tilde{\theta}_{tr}$ is better than $\tilde{\theta}_{[1,\dots,n]}$ as a whole, when (33) is satisfied. And (33) can be simplified to be (12). □

Appendix B. Proof of theorem 1

Proof. Assume estimators $\tilde{\theta}_{[1,\dots,n]}$ are unbiased with $b(\tilde{\theta}_k)=0$, $k=1,\dots,n$. Then we have

$$E[(\tilde{\theta}_k-\theta_k)^2]=Var(\tilde{\theta}_k),\quad k=[1,\dots,n], \tag{34}$$

and (29) can be simplified as

$$\sum_{k=1}^{n}E[(\tilde{\theta}_{tr}-\theta_k)^2]=\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)+\sum_{j=1}^{n}(\theta_j^2-\overline{\theta}^2). \tag{35}$$

By (34) and (35), we have

$$\frac{1}{n}\sum_{i=1}^{n}Var(\tilde{\theta}_i)+\sum_{j=1}^{n}(\theta_j^2-\overline{\theta}^2)\le\sum_{k=1}^{n}Var(\tilde{\theta}_k), \tag{36}$$

which is the necessary and sufficient condition that makes (10) satisfied when estimators $\tilde{\theta}_{[1,\dots,n]}$ are unbiased. And (36) can be simplified as (11). □

References

[1] M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: a review, ACM Sigmod Record 34 (2) (2005) 18–26.
[2] S. Muthukrishnan, Data streams: algorithms and applications, Foundations and Trends® in Theoretical Computer Science 1 (2) (2005) 117–236.
[3] S. Vucetic, Z. Obradovic, Classification on data with biased class distribution, in: ECML, 2001, pp. 527–538.
[4] N. Japkowicz, The class imbalance problem: a systematic study, Intelligent Data Analysis 6 (5) (2002) 429–449.
[5] R. Barandela, J. Sanchez, V. Garcıa, E. Rangel, Strategies for learning in class imbalance problems, Pattern Recognition 36 (3) (2003) 849–851.
[6] G. Batista, R. Prati, M. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explorations 6 (1) (2004) 20–29.
[7] A. Tsymbal, The problem of concept drift: definitions and related work, Technical Report, Department of Computer Science Trinity College, Dublin, Ireland, 2004.
[8] H. Wang, W. Fan, P. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: KDD, 2003, pp. 226–235.
[9] J. Kolter, M. Maloof, Dynamic weighted majority: an ensemble method for drifting concepts, The Journal of Machine Learning Research 8 (December) (2007) 2755–2790.
[10] D. Widyantoro, J. Yen, Relevant data expansion for learning concept drift from sparsely labeled data, IEEE Transactions on Knowledge and Data Engineering 17 (3) (2005) 401–412.
[11] H. Wang, J. Yin, J. Pei, P. Yu, J. Yu, Suppressing model overfitting in mining concept-drifting data streams, in: KDD, 2006, pp. 736–741.
[12] G. Forman, Tackling concept drift by temporal inductive transfer, in: SIGIR, 2006, pp. 252–259.
[13] P. Zhang, X. Zhu, Y. Shi, Categorizing and mining concept drifting data streams, in: KDD, 2008, pp. 812–820.
[14] X. Zhu, Semi-supervised learning literature survey, Technical Report 1530, Computer Sciences Department, University of Wisconsin-Madison, Wisconsin, USA, 2005.

[15] V. Vapnik, Estimation of Dependences Based on Empirical Data, Springer, New York, USA, 2006.
[16] T. Joachims, Transductive inference for text classification using support vector machines, in: ICML, 1999, pp. 200–209.
[17] R. El-Yaniv, D. Pechyony, Stable transductive learning, in: COLT, 2006, pp. 35–49.
[18] D. Boes, On the estimation of mixing distributions, The Annals of Mathematical Statistics 37 (1) (1966) 177–188.
[19] D. Kazakos, Recursive estimation of prior probabilities using a mixture, IEEE Transactions on Information Theory 23 (2) (1977) 203–211.
[20] G. Dattatreya, L. Kanal, Asymptotically efficient estimation of prior probabilities in multiclass finite mixtures, IEEE Transactions on Information Theory 37 (3) (1991) 482–489.
[21] G. McLachlan, J. Wiley, W. InterScience, Discriminant Analysis and Statistical Pattern Recognition, Wiley, New York, USA, 1992.
[22] G. Forman, Quantifying trends accurately despite classifier error and class imbalance, in: KDD, 2006, pp. 157–166.
[23] M. Saerens, P. Latinne, C. Decaestecker, Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure, Neural Computation 14 (1) (2002) 21–41.
[24] P. Latinne, M. Saerens, C. Decaestecker, Adjusting the outputs of a classifier to new a priori probabilities may significantly improve classification accuracy: evidence from a multi-class problem in remote sensing, in: ICML, 2001, pp. 298–305.
[25] D. Miller, S. Pal, Transductive methods for the distributed ensemble classification problem, Neural Computation 19 (3) (2007) 856–884.
[26] C. Yang, J. Zhou, Non-stationary data sequence classification using online class priors estimation, Pattern Recognition 41 (8) (2008) 2656–2664.
[27] S. Raudys, A. Jain, Small sample size effects in statistical pattern recognition: recommendations for practitioners, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (3) (1991) 252–264.
[28] S. Pan, Q. Yang, A Survey on Transfer Learning, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China, 2008.
[29] A. Ng, M. Jordan, On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes, in: NIPS, 2002, pp. 841–848.
[30] C. Bishop, Pattern Recognition and Machine Learning, Springer, New York, USA, 2006.
[31] J. Shao, Mathematical Statistics, Springer, New York, USA, 2003.
[32] W. Dai, Q. Yang, G. Xue, Y. Yu, Boosting for transfer learning, in: ICML, 2007, pp. 193–200.
[33] R. Raina, A. Battle, H. Lee, B. Packer, A. Ng, Self-taught learning: transfer learning from unlabeled data, in: ICML, 2007, pp. 759–766.
[34] W. Dai, Y. Chen, G. Xue, Q. Yang, Y. Yu, Translated learning: transfer learning across different feature spaces, in: NIPS, 2008, pp. 353–360.
[35] M. Sugiyama, S. Nakajima, H. Kashima, P. von Bunau, M. Kawanabe, Direct importance estimation with model selection and its application to covariate shift adaptation, in: NIPS, 2008, pp. 1433–1440.
[36] H. Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function, Journal of Statistical Planning and Inference 90 (2) (2000) 227–244.
[37] I. Csiszar, Information-type measures of difference of probability distributions and indirect observations, Studia Scientiarum Mathematicarum Hungarica 2 (1967) 299–318.
[38] J. Lin, Divergence measures based on the Shannon entropy, IEEE Transactions on Information Theory 37 (1) (1991) 145–151.
[39] S. Hettich, S. Bay, UCI KDD archive, 1999. [Online]. Available: 〈http://kdd.ics.uci.edu〉.

About the Author—ZHIHAO ZHANG was born in 1984. He received the B.E. degree from the Department of Control Science and Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006. He is currently pursuing the Ph.D. degree in the Department of Automation, Tsinghua University, Beijing, China.
His research interests are in pattern recognition, machine learning, data mining and intelligent information processing.

About the Author—JIE ZHOU was born in 1968. He received the B.S. and M.S. degrees from the Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively, and the Ph.D. degree from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology (HUST), Wuhan, China, in 1995.
From 1995 to 1997, he was a Postdoctoral Fellow with the Department of Automation, Tsinghua University, Beijing, China. Currently, he is a Full Professor with the Department of Automation, Tsinghua University. His research area includes pattern recognition, computer vision, and data mining. He has authored more than 100 papers in international journals and conferences. He is an associate editor for the International Journal of Robotics and Automation.
Dr. Zhou received the Best Doctoral Thesis Award from HUST, the First Class Science and Technology Progress Award from the Ministry of Education, China, and the Excellent Young Faculty Award from Tsinghua University in 1995, 1998, and 2003, respectively.