Improved TrAdaBoost and Its Application to Transaction Fraud Detection

Lutao Zheng, Guanjun Liu, *Senior Member, IEEE*, Chungang Yan, Changjun Jiang, Mengchu Zhou, *Fellow, IEEE*, and Maozhen Li, *Member, IEEE*

***Abstract*— AdaBoost is a boosting-based machine learning method under the assumption that the data in training and testing sets have the same distribution and input feature space. It increases the weights of those instances that are wrongly classified in a training process. However, the assumption does not hold in many real-world data sets. Therefore, AdaBoost is extended to transfer AdaBoost (TrAdaBoost) that can effectively transfer knowledge from one domain to another. TrAdaBoost decreases the weights of those instances that belong to the source domain but are wrongly classified in a training process. It is more suitable for the case that data are of different distribution. Can it be improved for some special transfer scenarios, e.g., the data distribution changes slightly over time? We find that the distribution of credit card transaction data can change with the changes in the transaction behaviors of users, but the changes are slow most of the time. These changes are yet important for detecting transaction fraud since they result in a so-called concept drift problem. In order to make TrAdaBoost more suitable for the abovementioned case, we, thus, propose an improved TrAdaBoost (ITrAdaBoost) in this article. It updates (i.e., increases or decreases) the weight of a wrongly classified instance in a source domain according to the distribution distance from the instance to a target domain, and the calculation of distance is based on the theory of reproducing kernel Hilbert space. We do a series of experiments over five data sets, and the results illustrate the advantage of ITrAdaBoost.**

***Index Terms*— Boosting learning, E-commerce, transaction fraud detection, transfer learning.**

I. Introduction

DATA mining and machine learning are successfully applied in many fields [65]. Traditional machine learning methods, such as AdaBoost, are based on such an assumption that the training and testing sets have the same data distribution and input feature space. They cannot achieve good performance when the assumption is invalid [51]. The assumption limits its applications since the data in many fields do not satisfy it [13]. Transfer learning is used to improve a learner by transferring information from a domain into another one [9], [27], [60], [65]. It breaks through the limitation of the abovementioned assumption and can then be successfully used in many applications, such as imbalanced data classification [3], [36], image recognition [29], [32], human activity prediction [23], [40], [69], fault detection [41], [72], multilanguage text classification [76], and recommender systems [15], [38], [39], [45], [49].

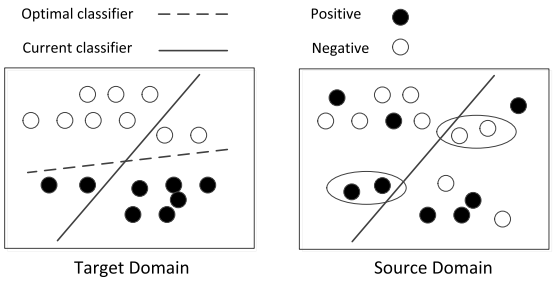


Fig. 1. Illustration of the factor ignored by TrAdaboost.

As an extension to AdaBoost [17], [68], TrAdaBoost [13] is an effective transfer learning method. It tries to filter out those instances in a source domain, which are very different from the instances in a target domain. If an instance in the source domain is wrongly classified in an iteration, then it is thought of as being dissimilar to the target domain, and then, its weight is decreased in the next iteration. On the contrary, AdaBoost is a boosting process with many weak learners. In each iteration, there are some instances that are wrongly classified because of the low accuracy of weak learners. Therefore, their influence is strengthened in the next iteration by increasing their weights. Zhang *et al.* [74] used TrAdaBoost to generate synthetic instances and then deal with a class imbalance problem.

TrAdaboost ignores different classification errors produced by a weak classifier in each iteration. However, an instance in a source domain wrongly classified by TrAdaBoost in an iteration possibly has the different distribution with instances in a target domain or the same/similar distribution with instances in a target domain. For the first case, its weight should indeed be decreased in the next iteration in order to weaken its influence. For the second one, however, it is unreasonable to decrease its weight. We should increase its weight in order to strengthen its influence. As shown in Fig. 1, there are two domains of data: source and target domains. The solid line represents the current classifier, and the dotted one represents the optimal

Manuscript received March 9, 2020; revised July 2, 2020; accepted August 10, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB2100801, and in part by the Fundamental Research Funds for the Central Universities of China under Grant 22120190198. *(Corresponding authors: Guanjun Liu; Changjun Jiang.)*

Lutao Zheng is with the Department of FinTech Research and Development, CaiTong Security Company Ltd., Hangzhou 310000, China (e-mail: zhenglutao103@163.com).

Guanjun Liu, Chungang Yan, Changjun Jiang, and Maozhen Li are with the Key Laboratory of the Ministry of Education for Embedded System and Service Computing, Tongji University, Shanghai 201804, China, and also with the Collaborative Innovation Center of Shanghai for Electronic Transactions and Information Service, Tongji University, Shanghai 201804, China (e-mail: liuguanjun@tongji.edu.cn; yanchungang@tongji.edu.cn; cjjiang@tongji.edu.cn; maozhen.li@brunel.ac.uk).

Mengchu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/TCSS.2020.3017013

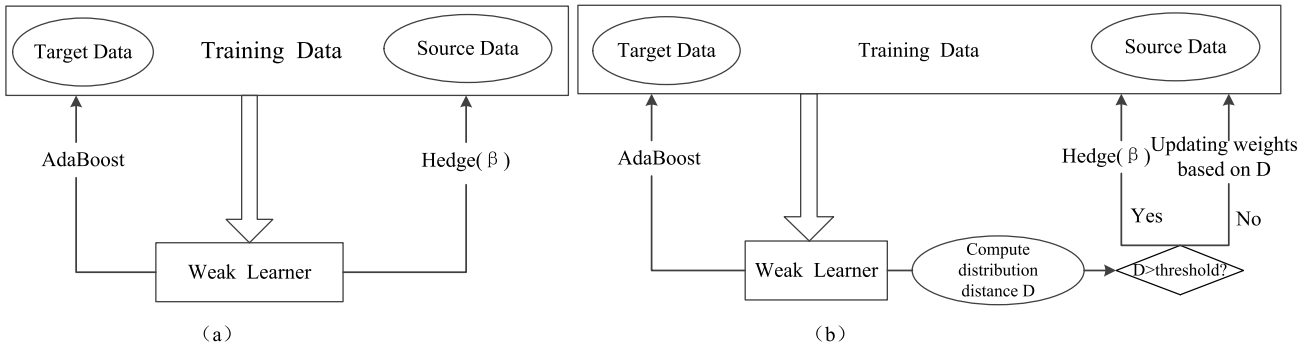


Fig. 2. Frameworks of (a) TrAdaBoost and (b) ITrAdaBoost.

classifier of the target domain. Both target and source domains have some instances that are wrongly classified by the current classifier. According to TrAdaBoost, the weights of the wrongly classified instances in the target domain are increased, and the weights of the wrongly classified instances in the source domain are decreased. However, since the instances in the circles in the source domain have a similar distribution with the instances in the target one, their weights should be increased but not decreased when they are wrongly classified.

Based on the abovementioned idea, we improve transfer AdaBoost (TrAdaBoost), named ITrAdaBoost, in order to fit some more complex data and, thus, enhance the classification performance. When an instance in the source domain is classified wrongly, we decrease its weight if it has a different distribution with the instances in the target domain. Otherwise, we increase its weight. It is obvious that a key question is how to evaluate the distribution similarity. In this article, we answer it based on distribution distance [20]. If the distance is greater than a threshold, then the weight is decreased just as TrAdaBoost does. If the distance is less than or equal to the threshold, then the weight is increased based on the value of the distance. Fig. 2 illustrates the frameworks of TrAdaBoost and ITrAdaBoost. The main contributions of this article are summarized as follows.

1) We use maximum mean discrepancy (MMD) in reproducing kernel Hilbert space (RKHS) to compute distribution distance and develop the algorithm of ITrAdaBoost.
2) We apply ITrAdaBoost to detect fraud transactions by which the concept drift problem can be solved well.
3) We conduct a series of experiments over four public data sets to demonstrate the superiority of ITrAdaBoost.

The rest of this article is organized as follows. Section II reviews the related work of TrAdaBoost. Section III recalls AdaBoost and TrAdaBoost. Section IV introduces distribution distance. ITrAdaBoost is presented in Section V. Section VI applies it to transaction fraud detection. Experiments and performance comparison are illustrated in Section VII. Finally, we conclude our work in Section VIII.

II. Related Work

There are many studies on transfer learning. Different from traditional machine learning, transfer learning permits data, which are from other domains, have a different distribution with testing data to take part in a training [43]. Pan and Yang [43] gave a comprehensive overview of transfer learning and introduced their applications, such as classification, regression, and clustering. In view of data labels, there are three categories of transfer learning: inductive [64], transductive [12], and unsupervised transfer learning methods [47]. Inductive one is suitable for the case in which there is not too much labeled data in the target domain. As for the transductive one, only the data in the source domain are labeled. It is obvious that unsupervised one does not require any data labels.

We can also categorize transfer learning methods into the following three cases: feature-selection-based [78], feature-representation-based, and weight-based [79]. The first one aims to find the common features between target and source domains and then uses them to transfer knowledge [12], [19], [26], [37], [80]. Dai *et al.* [12] proposed a coclustering-based method to enable knowledge and label transfer. In this method, features and labels of data are clustered simultaneously. Jiang and Zhai [26] proposed a two-stage framework to select features for the domain adaption problem. They think that those features, which have a high relevance with labels, should be assigned a high weight. Fang *et al.* [16] used a transfer learning method to learn a collaborative classification of different networks and tried to transfer some valuable knowledge in a given network to predict node labels in another new network.

The second one is to map the high-dimensional feature space of each domain into a low-dimensional one so that the data in target and source domains have the same distribution in the low-dimensional space [8], [42], [52], [53], [58]. The labeled data can be used in the source domain to train a classifier in this low-dimensional feature space and then predict test data in the target domain. The difference between this method and the feature-selection-based transfer learning methods is that the features derived by it are not in the original feature space. Blitzer *et al.* [8] proposed a structural learning algorithm to automatically induce the feature correspondences among different domains. Pan *et al.* [42] learned a low-dimensional latent feature space in which the distribution between the target and source domains is the same or similar. Then, a traditional machine learning model can be trained in this space. Gu and Zhou [22] proposed a framework to find a common feature subspace of target and source domains. Shao *et al.* [50] proposed a visual classification method based on transfer learning. This method maps the data in the target

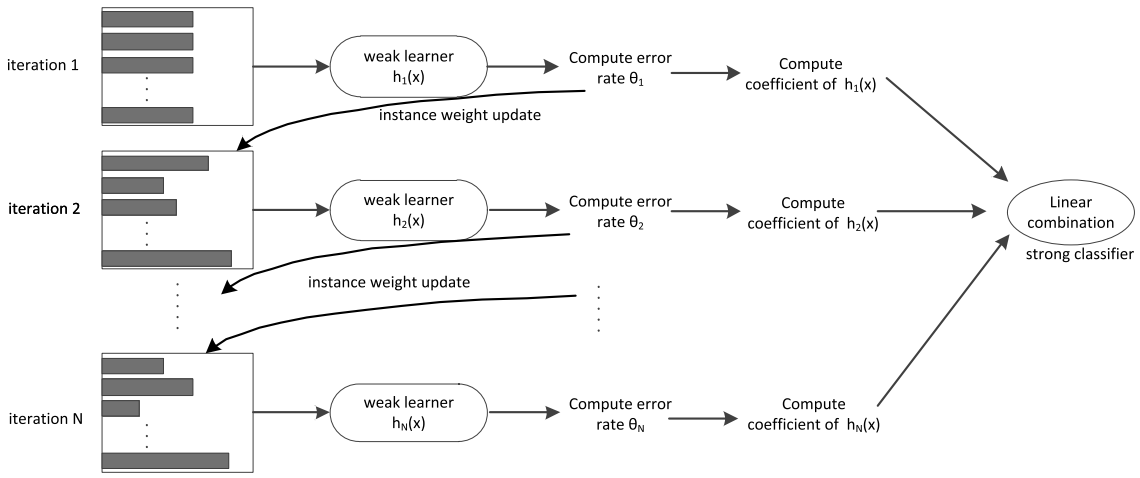


Fig. 3. Framework of AdaBoost.

and source domains to a generalized transfer subspace where the data in the target domain can be represented by a combination of the data in the source domain.

Sometimes, the distribution of instances in a target domain is different from the distribution of instances in a source domain [11], [33], [71], [77]. Therefore, some instances in the source domain are useless if not harmful [5]. For example, in the data set of our credit card transactions, some old transaction data may be outdated for predicting a future transaction behavior of a user, and then, the distribution of the transaction data in a period of time does not follow the same distribution in a latter period anymore. However, these data are of some similarity. How to increase the impact of the instances in the source domain for predicting the test data in the target domain? This problem can be solved by weight-based transfer learning. It is also called instance-based transfer learning that is to redistribute the weight of each instance in the source domain based on the difference between the source and target domains. In other words, we increase the weights of those instances in the source domain, which are useful for predicting the test data, and decrease the weights of those instances that are not. Tan *et al.* [6] selected useful unlabeled data from intermediate domains as a bridge to break the large distribution gap for transferring knowledge between two distant domains. Wang *et al.* [61] proposed an instance-based approach to improve deep transfer learning in a target domain. Dai *et al.* [13] applied AdaBoost [17] in transfer learning.

Our work shows that TrAdaBoost is not well suitable for the case where the data distribution in source and target domains is different but similar, e.g., the data of credit card transactions. Therefore, we propose ITrAdaBoost.

III. AdaBoost and TrAdaBoost Background

AdaBoost is a traditional machine learning method [17]. Its main idea is to train a set of weak learners through an iteration process. In each iteration, a weak learner is tweaked by the instances that are wrongly classified in the previous iteration. The weight of each instance is updated based on the classification result in the previous iteration. If an instance is wrongly classified in the previous iteration, then its weight is increased in the current iteration. In addition, every iteration produces a weak learner that also has a weight computed according to the error rate of the result of the iteration. Its final output is a weighted sum of all weak learners. Its framework is shown in Fig. 3. Zhang and Yang [73] proposed a general AdaBoost framework with a robust threshold mechanism and structural optimization that can well solve the regression problems.

TrAdaBoost [13] is an extension to AdaBoost. Its training set consists of two parts: training instances from a source domain and training instances from a target domain. Its basic idea is to filter out those instances in the source domain, which are dissimilar to those in the target domain. In TrAdaBoost, AdaBoost is still used for the instances in the target domain during the boosting process. However, for the instances in the source domain, if they are wrongly classified, their weights are decreased according to Hedge(β) [17] in order to minimize their influences. Especially, if an instance x in a source domain is wrongly classified, its weight is multiplied by $\beta^{|h_t(x)-c(x)|} \in (0, 1]$, where $h_t(x)$ is the predicted result of x by the weak learner and $c(x)$ is a real number representing the label of x. Fig. 2(a) illustrates its framework.

IV. Computation of Distribution Distance

A. MMD and RKHS

In our proposed method, we use MMD in RKHS to compute the distance between two distributions. In what follows, we give a simple introduction to them.

MMD is defined in terms of a particular function space, and the distance between two distributions can be computed based on it [20]. Let $X = \{x_1, x_2, \ldots, x_m\}$ and $Z = \{z_1, z_2, \ldots, z_n\}$ be two variable sets defined on the instance space with independently and identically distribution p and q, respectively. Let $E_X[f(x)]$ be the mean of the values of all $x \in X$ under function $f(x)$. $x \sim p$ represents that x is under the distribution p. Similarly, one can understand $E_Z[f(z)]$ and $z \sim q$. Let F be a class of functions such that $f \in F$ is continuous in the input space. MMD of p and q can be

computed as follows:

$$\Omega[F, p, q] = \sup_{f \in F} (E_X[f(x)] - E_Z[f(z)]). \tag{1}$$

The unbiased empirical estimate of MMD of X and Z can be computed as follows:

$$\Omega_u[F, X, Z] = \sup_{f \in F} \left(\frac{1}{m} \sum_{i=1}^{m} f(x_i) - \frac{1}{n} \sum_{j=1}^{n} f(z_j) \right). \tag{2}$$

It is obvious that the accuracy of MMD depends on the quality of function class F. Generally speaking, F should satisfy two properties [20]: 1) "rich" enough and 2) "restrictive" enough. When F is a set of unit balls in the characteristic RKHS [18], it satisfies them. We next give a simple introduce to RKHS [4], [7], [44], [63].

Let $\mathcal{H}$ is an RKHS of functions on X with kernel k such that

$$\langle f(\cdot), k(x, \cdot) \rangle = f(x)$$

and

$$\langle k(x, \cdot), k(x', \cdot) \rangle = k(x, x')$$

where $k(x, \cdot)$ is the feature mapping from a function f on X to its value at x as an inner product. Denote $\Phi(x) = k(x, \cdot)$ such that $k(x, y) = \langle \Phi(x), \Phi(y) \rangle$ [56]. Smola *et al.* [54] proved that if $E_X[k(x, x)] < \infty$, then $\mu[p_x] \in \mathcal{H}$ and $\mu[X] \in \mathcal{H}$, where $\mu[p_x] = E_X[\Phi(x)]$ and $\mu[X] = (1/m) \sum_{i=1}^{m} \Phi(x_i)$. Therefore, we can compute expectations and empirical means with respect to x and X via

$$\langle \mu[p_x], f \rangle = E_X[f(x)]$$

and

$$\langle \mu[X], f \rangle = \frac{1}{m} \sum_{i=1}^{m} f(x_i).$$

The abovementioned computation is to embed a probability distribution into RKHS [54]. Next, we introduce the computation of MMD in RKHS.

Gretton *et al.* [20] proved that when f is a unit ball in a universal RKHS, the necessary and sufficient condition of MMD $[f, p, q] = 0$ is $p = q$. Therefore, in what follows, we let f be a unit ball in a universal RKHS $\mathcal{H}$. We, thus, have

$$\begin{aligned}
\Omega[F, p, q] &= \sqrt{\mathrm{MMD}^2[F, p, q]} \\
&= \sqrt{\Big[\sup_{\|f\|_{\mathcal{H}} \le 1} \langle E_X[f(x)] - E_Z[f(z)] \rangle \Big]^2} \\
&= \sqrt{\Big[\sup_{\|f\|_{\mathcal{H}} \le 1} (\langle \mu[p_x], f \rangle - \langle \mu[p_z], f \rangle)_{\mathcal{H}} \Big]^2} \\
&= \sqrt{\Big[\sup_{\|f\|_{\mathcal{H}} \le 1} \langle \mu[p_x] - \mu[p_z], f \rangle_{\mathcal{H}} \Big]^2} \\
&= \sqrt{\|\mu[p_x] - \mu[p_z]\|_{\mathcal{H}}^2}.
\end{aligned} \tag{3}$$

Similarly, the unbiased empirical estimate of MMD of X and Z in RKHS can be computed as follows:

$$\begin{aligned}
\Omega_u[F, X, Z] &= \sqrt{\Omega_u^2[F, X, Z]} \\
&= \sqrt{\left[\sup_{\|f\|_{\mathcal{H}} \le 1} \left(\frac{1}{n} \sum_{i=1}^{n} f(x_i) - \frac{1}{m} \sum_{i=1}^{m} f(z_i) \right) \right]^2} \\
&= \Bigg[\sup_{\|f\|_{\mathcal{H}} \le 1} \Bigg(\frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \ne i}^{n} f(x_i) f(x_j) \\
&\qquad + \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \ne i}^{m-1} f(z_i) f(z_j) \\
&\qquad - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} f(x_i) f(z_j) \Bigg) \Bigg]^{\frac{1}{2}} \\
&= \Bigg[\frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \ne i}^{n-1} k(x_i, x_j) \\
&\qquad + \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \ne i}^{m-1} k(z_i, z_j) \\
&\qquad - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, z_j) \Bigg]^{\frac{1}{2}}.
\end{aligned} \tag{4}$$

B. Linear Computation Time and Optimal Kernel Choice

As we can see in (4), the computation complexity of MMD is $O(n^2)$. It is desirable to obtain $O(n)$ computation complexity at some affordable sacrifice of accuracy. Next, we introduce a linear-time method to compute an approximation of MMD [20].

Gretton *et al.* [20] proved that using the subsampling of the variable sets to compute MMD, we can obtain a linear-time computation complexity. For the sake of simplicity, let $m = n$, and define $\aleph = \lfloor n/2 \rfloor$. If $0 \le k(x_i, x_j) \le R$, and then

$$Pr_{X,Z}\left\{ \Omega_l^2(F, X, Z) - \Omega^2(F, X, Z) > t \right\} \le \exp\left(\frac{-t^2 \aleph}{8R^2} \right)$$

where

$$\Omega_l^2[F, X, Z] = \frac{1}{\aleph} \sum_{i=1}^{\aleph} h_k(v_i) \tag{5}$$

is an unbiased estimate of MMD and can be computed in a linear time, and

$$\begin{aligned}
h_k(v_i) = k(x_{2i-1}, x_{2i}) + k(z_{2i-1}, z_{2i}) \\
- k(x_{2i-1}, z_{2i}) - k(x_{2i}, z_{2i-1}).
\end{aligned}$$

We conduct a set of experiments to compare the computation time of Ω^2, Ω_l^2, and Kullback–Leibler (KL) divergence [30], as shown in Fig. 4. KL is also a kind of method to compute the distance between two probability distributions. We can find that as the number of instances increases, the growth rate of computation time of Ω^2 is much higher

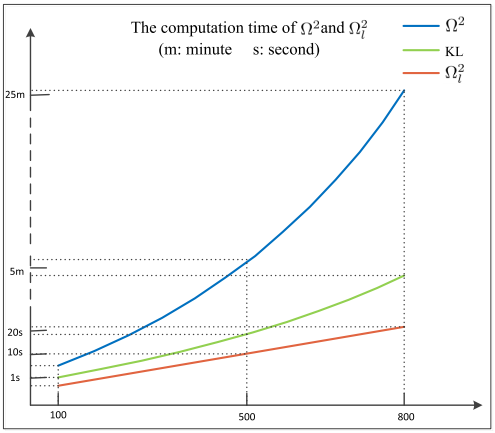


Fig. 4. Computation time of Ω^2, KL, and Ω_l^2.

than that of Ω_l^2, and the computation time of Ω_l^2 is smaller than that of KL. In addition, MMD accuracy depends on the choice of its kernel [21]. Next, we introduce an optimal kernel choice [21] for two-sample tests. Our goal is to select a proper kernel that can lead to the best performance on computing MMD.

Let $\{k_u\}_{u=1}^d$ be a set of kernel functions. The linear combination of these functions can be expressed as follows:

$$\mathcal{K} = \left\{ k \,\middle|\, k = \sum_{u=1}^{d} \beta_u k_u, \sum_{u=1}^{d} \beta_u = D, \beta_u \geq 0 \right\}$$

where $\forall u \in \{1, \ldots, d\}$, β_u is the coefficients of k_u, and $D > 0$ is the constraint of the sum of the coefficients. Our purpose is to find an optimal β that can lead to a high accuracy on computing MMD. Each $k \in \mathcal{K}$ is associated uniquely with an RKHS $\mathcal{H}$. Thus, the squared MMD can be written as

$$\eta_k(p, q) = \| \mu[p] - \mu[q] \|_{\mathcal{H}_k}^2 = \sum_{u=1}^{d} \beta_u \eta_u(p, q)$$

where $\eta_u(p, q) = \| \mu[p] - \mu[q] \|_{\mathcal{H}_{k_u}}^2$. The unbiased estimate of MMD can be written as

$$\widetilde{\eta}_k(p, q) = \sum_{u=1}^{d} \beta_u \widetilde{\eta}_u(p, q)$$

where $\widetilde{\eta}_u(p, q) = (1/\aleph) \sum_{i=1}^{d} h_u(\upsilon_i)$.

Based on the central limit theorem [48], if $0 < E(h_k^2) < \infty$, then

$$\frac{1}{\aleph}(\widetilde{\eta}_k(p, q) - \eta_k(p, q)) \to \mathcal{N}\left(0, 2\sigma_k^2\right)$$

where $\sigma_l^2 = Eh_k^2(\upsilon) - [E(h_k(\upsilon))]^2$. In terms of the kernel coefficients β, we can get

$$\sigma_k^2 = \beta^T Q_k \beta$$

where $Q_k = \mathrm{cov}(h_k)$ is the covariance matrix of h_k. Gretton *et al.* [20] proved that given an upper bound of the type-I error[1] α, the asymptotic probability of a type-II error is

$$P(\eta_k < t_{k,\alpha}) = \Phi\left(\Phi^{-1}(1-\alpha) - \frac{\eta_k(p, q)\sqrt{n}}{\sigma_k\sqrt{2}}\right)$$

where $t_{k,\alpha} = (1/\sqrt{n})\sigma_k\sqrt{2}\Phi^{-1}(1-\alpha)$. As Φ is monotonic, the type-II error is of an inverse relation with the ratio $\eta_k(p, q)\sigma_k^{-1}$. In practice, we cannot compute $\eta_k(p, q)$ and σ_k directly, but we can compute their empirical estimates $\widetilde{\eta}_k(p, q)$ and $\widetilde{\sigma}_k$. It has been proven that $\sup_{k\in\mathcal{K}} \widetilde{\eta}_k(\widetilde{\sigma}_{k,\lambda})^{-1}$ is convergent to $\sup_{k\in\mathcal{K}} \eta_k(p, q)\sigma_k^{-1}$ [21]. Therefore, the kernel minimizing type-II error can be defined as

$$\widetilde{k} = \arg\sup_{k\in\mathcal{K}} \widetilde{\eta}_k(\widetilde{\sigma}_{k,\lambda})^{-1}$$

where $\widetilde{\sigma}_{k,\lambda} = (\widetilde{\sigma}_k^2 + \lambda_n \| \beta \|_2^2)^{1/2}$ (λ_n is 10^{-4} in our experiments). Our purpose is to select a kernel $k = \sum_{u=1}^{d} \beta_u k_u \in \mathcal{K}$ that maximizes the ratio $\widetilde{\eta}_k(\widetilde{\sigma}_{k,\lambda})^{-1}$. Therefore, our purpose can become a quadratic program with a unique solution of

$$\min\left\{\widetilde{\sigma}_k^2 + \lambda_n \| \beta \|_2^2 : \beta^T \widetilde{\eta}_k = 1, \beta \geq 0\right\}. \tag{6}$$

After computing β, we can get a unique kernel k. For our data sets, we can compute their distribution distance according to the abovementioned theory. In Section V, we introduce ITrAdaBoost.

V. ITrAdaBoost

ITrAdaBoost is to find a classifier $H(x) : x \to y$ that has a high performance of classification on a testing set. When the training data that have the same distribution with the testing data are scarce, some data from another domain (or some outdated data in the same domain) can be used to enrich the training set in a target domain. Therefore, the training data of ITrAdaBoost come from both source and target domains. Generally, data distributions between the two domains are different.

Formally, we denote $X^T = \{x_1^T, x_2^T, \ldots, x_m^T\}$ and $X^S = \{x_1^S, x_2^S, \ldots, x_n^S\}$ as the training data instances coming from the target and source domains (without considering category labels of data), respectively. Y is denoted as the finite set of category labels. In fact, for each data instance $x \in X^T \cup X^S$, there is a category label corresponding to it, which is denoted as $c(x)$. Therefore, the training set (with category labels) is denoted as $\{(x, c(x)) | x \in X^T \cup X^S, c(x) \in Y\}$.

Next, we calculate the distance between an instance in the source domain and the whole target domain.

First, we fine-grainedly divide X^S into k parts: $S_1, S_2, \ldots$, and S_k, where $1 < k \leq n$. Note that the data set can be divided according to different factors. such as time. We then merge those parts that have the same distribution into a section, and finally, $S_1, S_2, \ldots$, and S_k are merged into $\mathcal{D}$ sections. Algorithm 1 describes a merge process. If the time complexity of training a classifier $h(x)$ is O_1 and the time complexity of computing the error rate on a section by $h(x)$ is O_2, then the time complexity of Algorithm 1 is $O(k \cdot (O_1 + k \cdot O_2))$ in the

[1]Type-I error rejects the assumption $p = q$ when the assumption is true, and the type-II error accepts the assumption when it is not true.

Algorithm 1 Merge Process

Input: $S_1, S_2, \cdots, S_k$, classifier $h(x)$, threshold T
Output: A group of sections

```
Q := {S_1, S_2, ..., S_k};
while Q ≠ ∅ do
    select an element S from Q;
    train a classifier h over S;
    Q := Q − {S};
    Q' := Q;
    while Q' ≠ ∅ do
        select an element S' from Q';
        Q' := Q' − {S'};
        compute the error rate θ of h(x) on S';
        if θ < T then
            S := S ∪ S';
            Q := Q − {S'};
        end
    end
    output a section S;
end
```

worst case. In our experiments, we select decision stump as a classifier. Its time complexity is $O(l*n^2*\log n)$, and the time complexity of each decision by decision stump is $O(\log n)$ [thus, the time complexity of computing the error rate on a section is averagely $O((n/k)\log n)$], where l is the dimension of each data sample and n is the number of data samples in a training set. Thus, the time complexity of Algorithm 1 is $O(k*(l*n^2*\log n + k\cdot(n/k)\log n)) = O(k*l*n^2*\log n)$.

For each section $X_i^S \subseteq X^S$ ($i \in \{1, 2, \ldots, \mathcal{D}\}$), we let $n_i = |X_i^S|$. Then, the distribution distance between X_i^S and X^T can be computed via (5) and is described as follows:

$$\begin{aligned} D^2\left(X_i^S, X^T\right) &= \mathrm{MMD}_l^2\left[F, X_i^S, X^T\right] \\ &= \frac{2}{n_i}\sum_{l=1}^{\frac{n_i}{2}} k\left(x_{2l-1}^S, x_{2l}^S\right) + \frac{2}{m}\sum_{j=1}^{\frac{m}{2}} k\left(x_{2j-1}^T, x_{2j}^T\right) \\ &\quad - \frac{2}{n_i}\sum_{l=1}^{\frac{n_i}{2}} k\left(x_{2l-1}^S, x_{2l}^T\right) - \frac{2}{m}\sum_{j=1}^{\frac{m}{2}} k\left(x_{2j-1}^T, x_{2j}^S\right) \end{aligned} \tag{7}$$

$\forall x \in X_i^S$, and we use $D(X_i^S, X^T)$ to represent the distance between x and X^T, i.e., $D(x, X^T) = D(X_i^S, X^T)$.

After computing the distribution distance, we introduce the mechanism of updating weights in ITrAdaBoost.

After the tth iteration, the weight of instance $x \in X_i^S \subseteq X^S$ can be computed as follows:

$$w_i^{t+1} = \begin{cases} w_i^t \cdot \beta^{|h_t(x)-c(x)|}, & D\left(X_i^S, X^T\right) > d \\ w_i^t \cdot [\beta\cdot(1+e^{D(x,X^T)})]^{-|h_t(x)-c(x)|} & \\ & D\left(X_i^S, X^T\right) \le d \end{cases} \tag{8}$$

where

$$\beta = \frac{\theta_t}{1-\theta_t}$$

and θ_t is the error rate produced by the weak learner on X^T in the tth iteration

$$\theta_t = \sum_{i=1}^{m} \frac{w_i^t \cdot \left|h_t\left(x_i^T\right) - c\left(x_i^T\right)\right|}{\sum_{i=1}^m w_i^t}. \tag{9}$$

The weight of instance x in X^T can be updated as follows:

$$w_i^{t+1} = w_i^t \cdot \beta^{-|h_t(x)-c(x)|}. \tag{10}$$

When an instance in a source domain is wrongly classified, we first decide whether its distribution is similar to the distribution of instances in a target domain. Based on (8), if the distribution distance is bigger than the threshold d, then we think that the distribution of data in the source domain is dissimilar to the distribution of the instance in the target domain, or else they are similar. Note that d is a superparameter, and its value is decided by experiments. If they are dissimilar, we decrease its weight in order to weaken its influence in the next iteration. In (8), the weight is decreased by multiplying a factor that is less than 1. If they are similar, this means that the error of the weak learner makes it wrongly classified, and thus, we increase its weight in the next iteration. In (8), the weight is increased by multiplying a factor that is greater than 1.

As shown in (10), the weights of instances in the target domain are updated in the same way as AdaBoost does.

Each iteration produces a weak learner and the coefficient of the tth weak learner can be computed as follows:

$$\alpha_t = \frac{1}{2}\log\frac{1-\theta_t}{\theta_t}. \tag{11}$$

Algorithm 2 realizes ITrAdaBoost. We first compute all distribution distances before executing Algorithm 2, and these distances are used in the step of updating weights in Algorithm 2. Therefore, the time complexity of Algorithm 2 is $O(N*(O_1+O_2))$, where N is the maximum number of iterations, O_1 is the time complexity of producing a weak classifier, and O_2 is the time complexity of computing the error rate in the target domain. The rest of the analysis is similar to Algorithm 1 and omitted here.

Theorem 1: Let $x \in X_i^S$, $z \in X_j^S$ ($i \ne j \le k$), $s \in X^T$, and d be the threshold of distribution distance. If $D(x, X^T) < D(z, X^T) \le d$, and x, z, and s are wrongly classified in the tth iteration, then

$$\frac{w_z^{t+1} - w_z^t}{w_z^t} < \frac{w_x^{t+1} - w_x^t}{w_x^t} < \frac{w_s^{t+1} - w_s^t}{w_s^t}.$$

Proof: Based on (8), if an instance x in a source domain is wrongly classified, then

$$\begin{aligned} w_x^{t+1} &= w_x^t \cdot [\beta\cdot(1+e^{D(x,X^T)})]^{-|h_t(x)-c(x)|} \\ &= w_x^t \cdot \beta^{-|h_t(x)-c(x)|} \cdot (1+e^{D(x,X^T)-d})^{-|h_t(x)-c(x)|} \end{aligned} \tag{12}$$

and $|h_t(x) - c(x)| = 1$. Therefore, we have

$$w_x^{t+1} = w_x^t \cdot \beta^{-1} \cdot (1+e^{D(x,X^T)-d})^{-1}$$

Algorithm 2 ITrAdaBoost

Input: X^T and X^S, weak learning algorithm, the maximum number of iterations N;
Output: The final classifier $H(x)$;

Initialize: The initial weight vector

$$w^1 = \left(w_1^1, w_2^1, \cdots, w_{m+n}^1\right),$$

w^1 can be initialized as $w_i^1 = \frac{1}{m+n}$, $i = 1, 2, \ldots, m+n$.

for $t = 1, t < N, t++$ **do**

  Set

$$p^t = \frac{w^t}{\sum_{i=1}^{m+n} w_i^t}$$

  Call the weak learning algorithm over X^T and X^S and then produce a base weak learner: $h_t(x) \to \{0, 1\}$;
  Calculate the error rate of $h_t(x)$ on X^T by Eq. 9;
  Calculate the coefficient of $h_t(x)$ by Eq. 11;
  Set $\beta = \frac{\theta_t}{1-\theta_t}$ and then update the weight of each training data as follows:
  if $x \in X^T$ **then**
    Update its weight by Eq. 10;
  else
    Update its weight by Eq. 8;
  end

end

Return a strong classifier:

$$H(x) = sign\left[\sum_{t=1}^{N} \alpha_t \cdot h_t(x)\right];$$

and

$$\frac{w_x^{t+1} - w_x^t}{w_x^t} = \beta^{-1} \cdot (1 + e^{D(x,X^T)-d})^{-1} - 1.$$

Since

$$(1 + e^{D(z,X^T)-d})^{-1} < (1 + e^{D(x,X^T)-d})^{-1} < 1$$

we have that

$$\frac{w_z^{t+1} - w_z^t}{w_z^t} < \frac{w_x^{t+1} - w_x^t}{w_x^t} < \frac{w_s^{t+1} - w_s^t}{w_s^t}.$$

□

Theorem 1 means that if an instance in a source domain is wrongly classified, the increase rate of its weight is less than the increase rate of the weights of the wrongly classified instances in a target domain. In addition, the more the similarity between the distribution of an instance in a source domain and that of a target domain, the larger the increase rate of its weight.

VI. Transaction Fraud Detection

In this section, we present the application of ITrAdaBoost to transaction fraud detection. With the popularization of online shopping, the events of transaction fraud are growing seriously, and thus fraud detection is necessary [25], [31], [34], [66]–[68], [75]. For transaction fraud detection, one simple approach is to train a model by using the history transaction data and then judge the legality of an incoming transaction via the model. This approach is based on an assumption that the training data and incoming data are of the same distribution [14], [46], [55]. However, the assumption does not hold in the real world since users' transaction behaviors generally change with time, leading to the so-called concept drift problem.

Concept drift [1], [62], [70] is a phenomenon that users change their transaction behavior over time due to an unstable external environment such as goods prices. In other words, their current transaction behavior pattern may be different from their historical pattern. Therefore, it is unreasonable to train a model using history transaction data directly. An alternative way is to update the training data using a window-sliding method with a forgetting mechanism [28]. However, it takes a basic assumption that the data distribution is changeable over time, i.e., data are become useless for predicting the current behavior when they are getting too old. However, the transaction behaviors of users do not necessarily completely drift. Some previously transaction behaviors may reappear after a period of time. Therefore, history data are still of potential value for training a model. The best way is to automatically select useful transaction data and filter out the useless ones when training a model.

In this article, we introduce a transaction fraud detection method based on ITrAdaBoost. First, the history transaction data are divided into two parts: source domain and target domain. The data in the target domain are thought of as having the same distribution with the incoming transaction data. For example, the data in the target domain can be the most recent transaction data or the transaction data that have a similar external environment with the current transaction. Transaction data in the source domain generally are generated before a relatively long period of time, and thus, many of them tend to have different distributions in comparison with the incoming transactions. The purpose of our fraud detection method based on ITrAdaBoost is to transfer knowledge from a source domain to a target one by increasing the influence of the useful data and decreasing the influence of the valueless data. Our method of computing distance can better evaluate the usefulness of data.

The framework of our fraud detection method is shown in Fig. 5. $\text{Case}_1, \ldots$, and Case_n represent n different situations of concept drift. For example, we can use the transaction data produced in the most recent month or week as the target domain and the data produced in other months or weeks as the source domain. If we want to predict the transaction behavior of users at weekends, we can use the transaction data produced in all previous weekends as the target domain and others in the working days as the source domain. Later, we illustrate our experiments on real transaction data set coming from a financial company in China.

VII. Performance Evaluation

In this section, we evaluate the performance of ITrAdaBoost through from two aspects. The first one uses four public

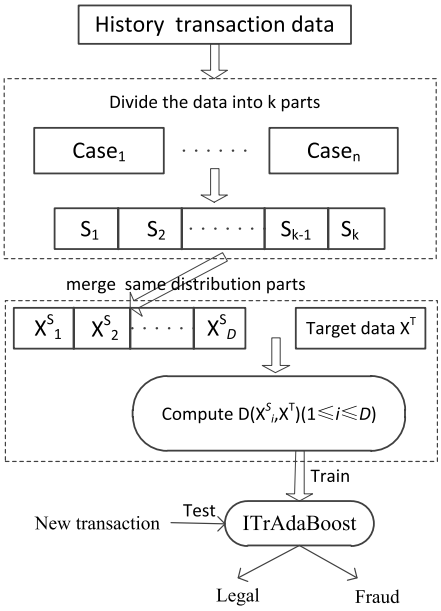


Fig. 5. Framework of fraud detection by ITrAdaBoost. Note that $D(-,-)$ means the function of computing distance and D means the number of sections yielded by Algorithm 1.

data sets, and the second one uses our real transaction data. In our experiments, the weak learner is a decision stump, i.e., a simple classifier consisting of a one-level decision tree [24]. Steinwart [57] proved that the Gaussian and Laplace RKHSs are universal. Therefore, Gaussian and Laplace kernel functions form our kernel function set K.

A. Experiment I

We select four data sets from Newsgroups[2] that satisfied the transfer learning situation. Newsgroups is a text data set with hierarchical structures consisting of seven top categories. Each top category is also an individual data set and has several subcategories, and each subcategory contains 1000 texts. In order to make the data set satisfy the transfer learning situation, we follow the method in [13] to split and revise the data set. First, the classification problem is defined in the top categories. We choose two top categories of text data sets for classification, and in each top category, we use one subcategory as the target domain and the other one as the source domain. Note that our test set comes from the target domain, and the remainders are used to train.

Especially, *rec* and *talk* are two top categories in Newsgroups. Therefore, data set Base-Guns can be selected as follows: we choose subcategory *rec.sport.baseball* from *rec* and subcategory *talk.politics.guns* from *talk* as a target domain. The source domain also consists of two kinds of data: one is subcategory *rec.sport.hockey* from *rec* and the other is subcategory *talk.politics.misc* from *talk*.

Data set Motor-Misc can be selected as follows: we choose subcategory *rec.motorcyles* from *rec* and subcategory *talk.religion.misc* from *talk* as a target domain. The source domain also consists of two kinds of data: one is the subcategory *rec.autos* from *rec* and the other is subcategory *talk.politics.misc* from *talk*.

[2]http://people.csail.mit.edu/jrennie/20Newsgroups/

TABLE I
ACCURACY OF ITRADABOOST ON DATA SET BASE-GUNS WITH DIFFERENT d'S AND DIFFERENT T'S

		P=5%	P=10%	P=20%	P=30%	P=40%
T=0.15	$d=0.08$	0.785	0.823	0.857	0.871	0.916
	$d=0.07$	0.783	0.836	0.861	0.892	0.926
	$d=0.06$	0.781	0.844	0.863	0.905	0.937
	$d=0.05$	0.761	0.829	0.845	0.874	0.923
	$d=0.04$	0.742	0.832	0.841	0.886	0.911
T=0.20	$d=0.08$	0.783	0.824	0.856	0.881	0.926
	$d=0.07$	0.783	0.823	0.857	0.882	0.924
	$d=0.06$	0.783	0.840	0.859	0.884	0.927
	$d=0.05$	0.754	0.814	0.846	0.874	0.913
	$d=0.04$	0.742	0.812	0.841	0.880	0.903
T=0.25	$d=0.08$	0.783	0.824	0.856	0.881	0.916
	$d=0.07$	0.779	0.816	0.842	0.881	0.919
	$d=0.06$	0.780	0.836	0.861	0.882	0.926
	$d=0.05$	0.765	0.831	0.845	0.872	0.917
	$d=0.04$	0.734	0.814	0.831	0.870	0.906

Data set Base-Misc can be selected as follows: we choose subcategory *rec.sport.baseball* from *rec* and subcategory *talk.religion.misc* from *talk* as a target domain. The source domain also consists of two categories of data: one is the subcategory *rec.sport.hockey* from *rec* and the other is subcategory *talk.politics.misc* from *talk*.

Data set Motor-Guns can be selected as follows: we choose subcategory *rec.motorcyles* from *rec* and subcategory *talk.politic.guns* from *talk* as a target domain. The source domain also consists of two categories of data: one is the subcategory *rec.autos* from *rec* and the other is subcategory *talk.politics.misc* from *talk*.

We first test the performance of ITrAdaBoost and set parameter d on each data set.

For each data set, we divide source data into D parts by considering three different thresholds of error rate. Table I shows the results of ITrAdaBoost on data set Base-Guns over each combination of parameters. The threshold T of error rate is set to 0.15, 0.2, and 0.25, respectively. From it, we can conclude that the smaller the value of T, the higher the performance of ITrAdaBoost. Furthermore, we select data from the target domain as the testing ones, and the remainders take part in the training. Table I shows the experimental results of the accuracy of ITrAdaBoost with different d's and different percentages of training and testing data in the target domain. Here, d is set by 0.04, 0.05, 0.06, 0.07, and 0.08, respectively. P is the percentage of data in the target domain used for training is set by 5%, 10%, 20%, 30%, and 40%. From Table I, we can see that the bigger the number of P, the higher the accuracy. When $d = 0.06$, the method has the highest accuracy. When $d = 0.06$ and $T = 0.15$, ITrAdaBoost performs the best on data set Base-Guns. To save space, we do not list the results of ITrAdaBoost on the other three data sets over each combination of parameters. When $d = 0.06$ and $T = 0.15$, ITrAdaBoost performs the best on data set Motor-Misc. When $d = 0.07$ and $T = 0.15$, ITrAdaBoost performs best on data set Base-Misc. When $d = 0.06$ and $T = 0.15$, ITrAdaBoost performs the best on data set Motor-Guns.

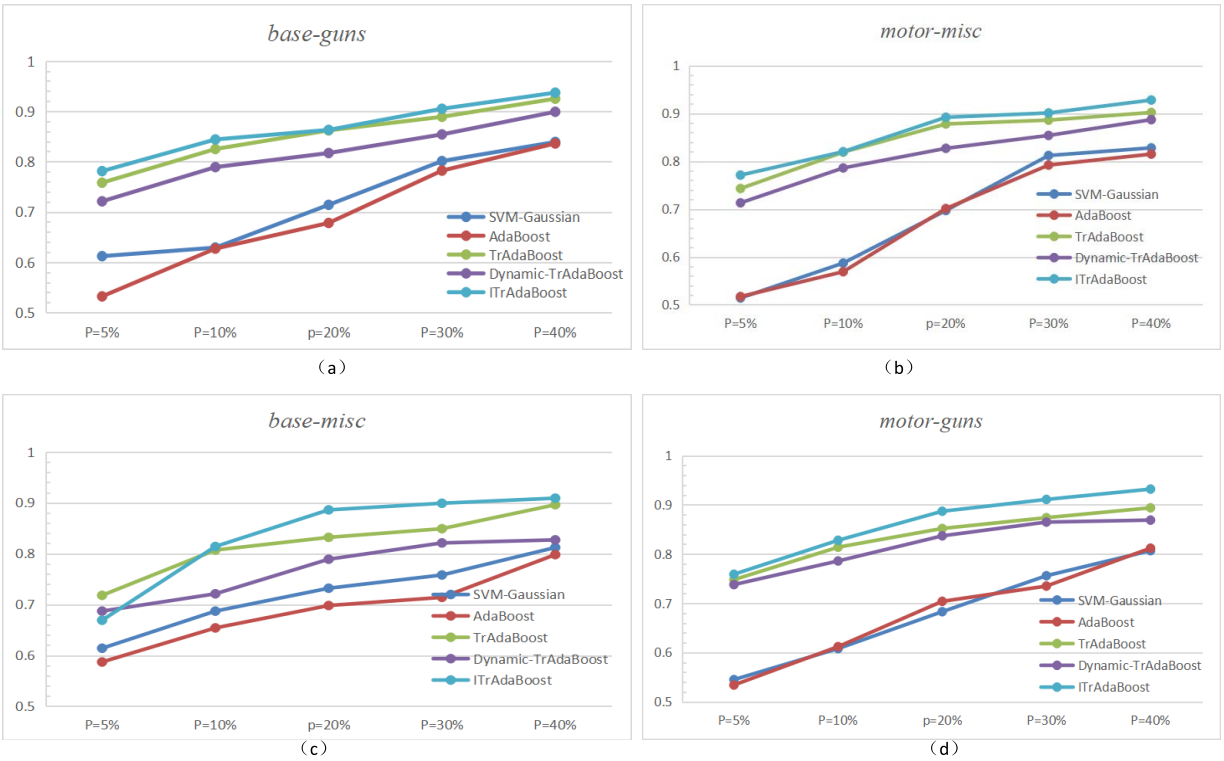


Fig. 6. Accuracy of five methods on five different transfer situations of four data sets. (a) Accuracy values on dataset base-guns. (b) Accuracy values on dataset motor-misc. (c) Accuracy values on dataset base-misc. (d) Accuracy values on dataset motor-guns.

Next, we compare ITrAdaBoost with the best parameters combination with the other four methods, including TrAdaBoost and AdaBoost on the four data sets. As shown in Fig. 6, SVM-Gaussian is a support vector machine [10] with a Gaussian kernel, and Dynamic-TrAdaBoost is an instance-based transfer learning method proposed in [2].

Fig. 6 shows the comparison result of the accuracy of the five methods on the four data sets with five different proportions. As shown in Fig. 6, their results become better when P increases, and the performance of AdaBoost and SVM-Gaussian is inferior to others. There are two main reasons: 1) there is not too much training data that have the same distribution as the test data does and 2) AdaBoost and SVM-Gaussian use the source data to train a model directly. ITrAdaBoost is slightly better than TrAdaBoost and Dynamic-TrAdaBoost.

B. Experiment II

In this section, we conduct the experiments on a real transaction data set provided by a financial company in China. The data are produced in six continuous months. In our experiment, we consider two types of concept drift: types I and II. Type I means that a user's transaction behaviors change over time. In this case, we split the transaction data as follows: the data in the earliest three months form the source domain, the data in the latest month as the testing set and the data in the other two months form the target domain. Type II is that users' transaction behaviors are different in weekdays and weekends. In this case, we select all the transaction data on weekends in the earliest three months as the target domain, all the transaction data on weekends in the latest three months as the testing set, and all the transaction data in weekdays as the source domain. Table II presents the number of fraudulent and legal transactions in these experiments.

TABLE II
NUMBER OF LEGAL AND FRAUD TRANSACTION

Concept drift	Training set				Testing set	
	Source domain		Target domain		legal	fraud
	legal	fraud	legal	fraud		
Type I	90680	1037	23320	283	27500	350
Type II	98750	1053	20950	317	21800	300

Next, we compare the performances of the five methods in the two types of concept drifts. We first divide the source domain into D parts based on Algorithm I, and we also consider three different thresholds, as shown in Tables III and IV, where T represents the threshold of an error rate that is calculated via F1. We can see from the two tables that the smaller the value of T, the higher the performance of ITrAdaBoost. The reason is that T influences the merging results of Algorithm 1: the smaller the value of T, the more the similar data in every section obtained by Algorithm 1, and thus, the better the performance of our method. However, we cannot let T be too small because a too small value cannot make Algorithm 1 take effect on emerging. The proportion of legal and fraudulent transactions is very unbalanced in our data; we use four metrics to evaluate their performances: recall, precision, F1, and AUC. Recall measures the accuracy

TABLE III

PERFORMANCE OF ITRADABOOST FOR TYPE-I CONCEPT DRIFT WITH DIFFERENT d'S AND T'S

		d=0.14	d=0.18	d=0.22	d=0.26	d=0.3
T=0.60	$F1$	0.329	0.482	0.428	0.347	0.322
	$recall$	0.32	0.66	0.58	0.52	0.52
	$precision$	0.34	0.38	0.34	0.26	0.20
	AUC	0.76	0.82	0.78	0.74	0.72
T=0.65	$F1$	0.310	0.466	0.423	0.369	0.332
	$recall$	0.30	0.66	0.56	0.48	0.46
	$precision$	0.32	0.36	0.34	0.30	0.26
	AUC	0.74	0.81	0.77	0.72	0.71
T=0.68	$F1$	0.34	0.498	0.453	0.396	0.354
	$recall$	0.34	0.66	0.56	0.52	0.48
	$precision$	0.34	0.40	0.38	0.32	0.28
	AUC	0.75	0.81	0.78	0.72	0.70

TABLE IV

PERFORMANCE OF ITRADABOOST FOR TYPE-II CONCEPT DRIFT WITH DIFFERENT d'S AND T'S

		d=0.13	d=0.17	d=0.21	d=0.25	d=0.29
T=0.60	$F1$	0.32	0.416	0.500	0.398	0.336
	$recall$	0.32	0.46	0.62	0.48	0.42
	$precision$	0.32	0.38	0.42	0.34	0.28
	AUC	0.73	0.79	0.81	0.74	0.71
T=0.65	$F1$	0.309	0.388	0.459	0.384	0.343
	$recall$	0.30	0.42	0.58	0.44	0.40
	$precision$	0.32	0.36	0.38	0.34	0.30
	AUC	0.71	0.77	0.80	0.73	0.71
T=0.68	$F1$	0.289	0.396	0.494	0.404	0.350
	$recall$	0.30	0.44	0.60	0.46	0.42
	$precision$	0.28	0.36	0.42	0.36	0.30
	AUC	0.71	0.78	0.80	0.75	0.74

on the fraudulent transactions. Precision gives the accuracy of transactions predicted as being fraudulent. F1 gives the harmonic mean of precision and recall. AUC measures the area under the ROC curve and is independent of specific classification cutoff values. The performance of ITrAdaBoost on type I is presented in Table III, and the best results are obtained when $d = 0.18$. As we can see from Table IV, ITrAdaBoost has the best performance on type II when $d = 0.21$. Therefore, d is set by 0.18 and 0.21 for types I and II, respectively. Note that for an imbalanced data set, accuracy is not suitable to measure the performance of a classification algorithm [36]. Thus, we cannot use it to determine the optimal values of these superparameters.

Figs. 7 and 8 present the results of the five methods for types I and II, respectively. In the iteration process of TrAdaBoost, some instances in the source domain are wrongly classified by a weaker classifier but not caused by the different distribution. However, TrAdaBoost does not consider this case. As a result, the performance of TrAdaBoost is not ideal. ITrAdaBoost outperforms AdaBoost and SVM-Gaussian, which means that the former can deal with the concept drift problem better than them.

Note that TrAdaboost is suitable for the scenario that the data distribution of source and target domains is similar. In other words, data are from two different entities, but these two entities are closely related, as shown by the examples in experiment I. However, when the data in source and target

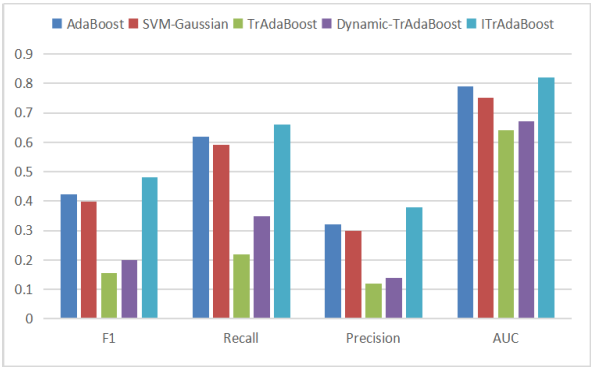


Fig. 7. Comparison of five methods for type-I concept drift.

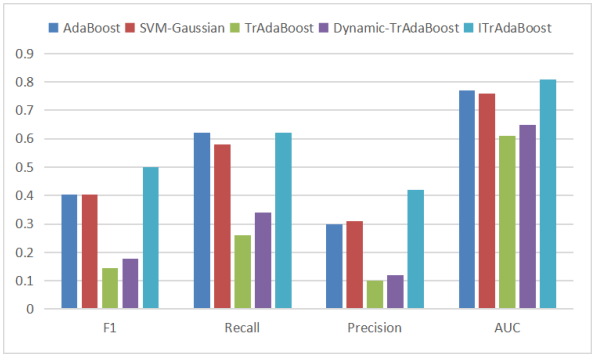


Fig. 8. Comparison of five methods for type-II concept drift.

domains are very similar, e.g., they are from the same entity (as shown by the examples in experiment II), TrAdaBoost possibly causes some negative transfer [43].

VIII. CONCLUSION

This article proposes ITrAdaBoost that improves the transfer learning method TrAdaBoost [13]. ITrAdaBoost considers the weights of wrongly classified instances more comprehensively. The computation of the distribution distance between the source domain and the target one is the foundation of our method that is based on the theory of RKHS. We also apply ITrAdaBoost to deal with a concept drift problem in transaction fraud detection. The experiments show that ITrAdaBoost outperforms TrAdaBoost, Dynamic-TrAdaBoost, AdaBoost, and SVM-Gaussian on the performance of classification. However, the computation time of our method is higher compared with others because we spend too much time on the computation of distribution distances. Therefore, our future work plans to decrease the computation complexity of ITrAdaBoost. This can be helped by deciding the best superparameter setting through the methods in [35]. We also plan to consider other weak classifiers in ITradaBoost, so as to verify if ITrAdaBoost is of excellent generalization ability. In addition, we should extend ITrAdaBoost based on feature-representation, which can better deal with the difference of the feature spaces of source and target domains. We also plan to study the interpretation of ITrAdaBoost since this problem is very important for fraud detection [59].

ACKNOWLEDGMENT

The authors would like to thank editors and the three reviewers for their constructive comments.

REFERENCES

[1] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *J. Netw. Comput. Appl.*, vol. 68, pp. 90–113, Jun. 2016.

[2] S. Al-Stouhi and C. K. Reddy, "Adaptive boosting for transfer learning using dynamic updates," in *Proc. 2011th Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2011, pp. 60–75.

[3] S. Al-Stouhi and C. K. Reddy, "Transfer learning for class imbalance problems with inadequate data," *Knowl. Inf. Syst.*, vol. 48, no. 1, pp. 201–228, Jul. 2016.

[4] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, no. 3, pp. 337–404, 2003.

[5] A. Asgarian *et al.*, "A hybrid instance-based transfer learning method," 2018, pp. 1–7, *arXiv:1812.01063*. [Online]. Available: http://arxiv.org/abs/1812.01063

[6] B. Tan, Y. Zhang, S. J. Pan, and Q. Yang, "Distant domain transfer learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 2604–2610.

[7] A. Berlinet and C. Thomas-Agnan, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. New York, NY, USA: Springer, 2004.

[8] J. Blitzer, R. Mcdonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2006, pp. 120–128.

[9] G. Cai, Y. Wang, L. He, and M. Zhou, "Unsupervised domain adaptation with adversarial residual transform networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 3073–3086, Aug. 2020, doi: 10.1109/TNNLS.2019.2935384.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[11] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3730–3739.

[12] W. Dai, G. R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2007, pp. 210–219.

[13] W. Dai, Q. Yang, G. R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 193–200.

[14] V. Dheepa and R. Dhanapal, "Behavior based credit card fraud detection using support vector machines," *ICTACT J. Soft Comput.*, vol. 2, no. 4, pp. 391–397, Jul. 2012.

[15] Z. Ding, X. Li, C. Jiang, and M. Zhou, "Objectives and state-of-the-art of location-based social network recommender systems," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–28, 2018.

[16] M. Fang, J. Yin, and X. Zhu, "Transfer learning across networks for collective classification," in *Proc. IEEE 13th Int. Conf. Data Mining*, Dec. 2013, pp. 161–170.

[17] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learn. Theory*, 1995, pp. 23–37.

[18] K. Fukumizu, "Kernel measures of conditional dependence," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, no. 1, 2009, pp. 167–204.

[19] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2030–2096, 2017.

[20] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 723–773, 2012.

[21] A. Gretton *et al.*, "Optimal kernel choice for large-scale two-sample tests," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1205–1213.

[22] Q. Gu and J. Zhou, "Learning the shared subspace for multi-task clustering and transductive transfer classification," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2009, pp. 159–168.

[23] M. Harel and S. Mannor, "Learning from multiple outlooks," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, 2011, pp. 401–408.

[24] W. Iba and P. Langley, "Induction of one-level decision trees," in *Proc. Int. Workshop Mach. Learn.*, 1992, pp. 233–240.

[25] C. Jiang, J. Song, G. Liu, L. Zheng, and W. Luan, "Credit card fraud detection: A novel approach using aggregation strategy and feedback mechanism," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3637–3647, Oct. 2018.

[26] J. Jiang and C. X. Zhai, "A two-stage approach to domain adaptation for statistical classifiers," in *Proc. Res. Collection School Inf. Syst.*, 2007, pp. 401–410.

[27] Q. Kang, S. Yao, M. Zhou, K. Zhang, and A. Abusorrah, "Enhanced subspace distribution matching for fast visual domain adaptation," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 4, pp. 1047–1057, Aug. 2020, doi: 10.1109/TCSS.2020.3001517.

[28] R. Klinkenberg, I. Renz, and D. B. Ag, "Adaptive information filtering: Learning in the presence of concept drifts," AAAI, Menlo Park, CA, USA, Tech. Rep. WS-98-05, 1998, pp. 33–40.

[29] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 1785–1792.

[30] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[31] Z. Li, G. Liu, and C. Jiang, "Deep representation learning with full center loss for credit card fraud detection," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 2, pp. 569–579, Apr. 2020.

[32] J. J. Lim, R. R. Salakhutdinov, and A. Torralba, "Transfer learning by borrowing examples for multiclass object detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 118–126.

[33] D. Lin, X. An, and J. Zhang, "Double-bootstrapping source data selection for instance-based transfer learning," *Pattern Recognit. Lett.*, vol. 34, no. 11, pp. 1279–1285, Aug. 2013.

[34] G. Liu and C. Jiang, "Behavioral equivalence of security-oriented interactive systems," *IEICE Trans. Inf. Syst.*, vol. E99.D, no. 8, pp. 2061–2068, 2016.

[35] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–604, Feb. 2019.

[36] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 3, pp. 703–715, May 2019.

[37] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," 2015, pp. 1–9, *arXiv:1502.02791*. [Online]. Available: http://arxiv.org/abs/1502.02791

[38] W. Luan, G. Liu, C. Jiang, and M. Zhou, "MPTR: A maximal-marginal-relevance-based personalized trip recommendation method," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 11, pp. 3461–3474, Nov. 2018.

[39] X. Luo, M. Zhou, S. Li, and M. Shang, "An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2011–2022, May 2018.

[40] Z. Ma, W. Yu, X. Zhai, and M. Jia, "A complex event processing-based online shopping user risk identification system," *IEEE Access*, vol. 7, pp. 172088–172096, 2019.

[41] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous defect prediction," *IEEE Trans. Softw. Eng.*, vol. 44, no. 9, pp. 874–896, Sep. 2018.

[42] S. Pan, J. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction," in *Proc. AAAI Conf. Artif. Intell.*, 2008, pp. 677–682.

[43] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[44] V. Paulsen and M. Raghupathi, *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces*. Cambridge, U.K.: Cambridge Univ. Press, 2016.

[45] P. Hao, G. Zhang, L. Martinez, and J. Lu, "Regularizing knowledge transfer in recommendation with tag-inferred correlation," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 83–96, Jan. 2019.

[46] C. Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," 2010, pp. 1–14, *arXiv:1009.6119*. [Online]. Available: http://arxiv.org/abs/1009.6119

[47] S. Samanta, A. Selvan, and S. Das, "Cross-domain clustering performed by transfer of knowledge across domains," in *Proc. 4th Nat. Conf. Comput. Vis., Pattern Recognit., Image Process. Graph.*, 2013, pp. 1–4.

[48] R. Serfling, *Approximation Theorems of Mathematical Statistics*. Hoboken, NJ, USA: Wiley, 2009.

[49] M. Shang, X. Luo, Z. Liu, J. Chen, Y. Yuan, and M. Zhou, "Randomized latent factor model for high-dimensional and sparse matrices from industrial applications," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 1, pp. 131–141, Jan. 2019.

[50] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," *Int. J. Comput. Vis.*, vol. 109, nos. 1–2, pp. 74–93, Aug. 2014.

[51] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Stat. Planning Inference*, vol. 90, no. 2, pp. 227–244, Oct. 2000.

[52] S. Si, D. Tao, and K.-P. Chan, "Evolutionary cross-domain discriminative Hessian eigenmaps," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 1075–1086, Apr. 2010.

[53] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 7, pp. 929–942, Jul. 2010.

[54] A. Smola, A. Gretton, L. Song, and B. Scholkopf, *A Hilbert Space Embedding for Distributions*. 2007, pp. 40–41.

[55] S. Sorournejad, Z. Zojaji, R. E. Atani, and A. H. Monadjemi, "A survey of credit card fraud detection techniques: Data and technique oriented perspective," 2016, pp. 1–26, *arXiv:1611.06439*. [Online]. Available: http://arxiv.org/abs/1611.06439

[56] I. Steinwart and A. Christmann, *Support Vector Machines*. Cham, Switzerland: Springer, 2008.

[57] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *J. Mach. Learn. Res.*, vol. 2, no. 1, pp. 67–93, Mar. 2002.

[58] X. Tian, D. Tao, and Y. Rui, "Sparse transfer learning for interactive video search reranking," *ACM Trans. Multimedia Comput., Commun., Appl.*, vol. 8, no. 3, pp. 1–19, Jul. 2012.

[59] Y. Tian and G. Liu, "MANE: Model-agnostic non-linear explanations for deep learning model," in *Proc. IEEE World Congr. Services (SERVICES)*, Beijing, China, Oct. 2020, p. 4.

[60] G. Wang, J. Qiao, J. Bi, W. Li, and M. Zhou, "TL-GDBN: Growing deep belief network with transfer learning," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 847–885, Oct. 2019.

[61] T. Wang, J. Huan, and M. Zhu, "Instance-based deep transfer learning," 2018, pp. 367–375.

[62] X. Wang, Q. Kang, M. Zhou, and S. Yao, "A multiscale concept drift detection method for learning from data streams," in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2018, pp. 786–790.

[63] E. Wegman, *Reproducing Kernel Hilbert Spaces*. Hoboken, NJ, USA: Wiley, 2004.

[64] W. Fengmei, Z. Jianpei, C. Yan, and Y. Jing, "FSFP: Transfer learning from long texts to the short," *Appl. Math. Inf. Sci.*, vol. 8, no. 4, pp. 2033–2040, Jul. 2014.

[65] K. Weiss, T. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, pp. 9–17, 2016.

[66] H. Wu and G. Liu, "A hybrid model on learning cross features for transaction fraud detection," in *Proc. 19th Ind. Conf. Data Mining*, 2019, pp. 88–102.

[67] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, and C. Jiang, "Random forest for credit card fraud detection," in *Proc. IEEE 15th Int. Conf. Netw., Sens. Control (ICNSC)*, Mar. 2018, pp. 1–6.

[68] C. Yang, G. Liu, C. Yan, and C. Jiang, "A clustering-based flexible weighting method in adaboost and its application to transaction fraud detection," *Sci. China Inf. Sci.*, to be published, doi: 10.1007/s11432-019-2739-2.

[69] W. Yu, Z. Ding, L. Liu, X. Wang, and R. D. Crossley, "Petri net-based methods for analyzing structural security in e-commerce business processes," *Future Gener. Comput. Syst.*, vol. 109, pp. 611–620, Aug. 2020.

[70] J. Zhang, Z. Wei, Z. Yan, M. Zhou, and A. Pani, "Online change-point detection in sparse time series with application to online advertising," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 6, pp. 1141–1151, Jun. 2019.

[71] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 819–827.

[72] P. Zhang, S. Shu, and M. Zhou, "An online fault detection model and strategies based on SVM-grid in clouds," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 2, pp. 445–456, Mar. 2018.

[73] P.-B. Zhang and Z.-X. Yang, "A novel AdaBoost framework with robust threshold and structural optimization," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 64–76, Jan. 2018.

[74] X. Zhang, Y. Zhuang, W. Wang, and W. Pedrycz, "Transfer boosting with synthetic instances for class imbalanced object recognition," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 357–370, Jan. 2018.

[75] L. Zheng, G. Liu, C. Yan, and C. Jiang, "Transaction fraud detection based on total order relation and behavior diversity," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 3, pp. 796–806, Sep. 2018.

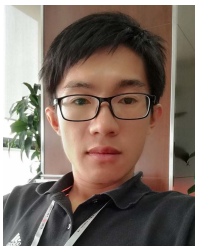
[76] J. T. Zhou, S. Pan, I. Tsang, and Y. Yan, "Hybrid heterogeneous transfer learning through deep learning," in *Proc. 8th AAAI Conf. Artif. Intell.*, 2014, pp. 2213–2219.

[77] S. Zhou, G. Schoenmakers, E. Smirnov, R. Peeters, K. Driessens, and S. Chen, "Largest source subset selection for instance transfer," in *Proc. Asian Conf. Mach. Learn.*, 2016, pp. 423–438.

[78] S. Zhou, E. N. Smirnov, G. Schoenmakers, and R. Peeters, "Conformal decision-tree approach to instance transfer," *Ann. Math. Artif. Intell.*, vol. 81, nos. 1–2, pp. 85–104, Oct. 2017.

[79] F. Z. Zhuang, P. Luo, Q. He, and Z. Shi, "Survey on transfer learning research," *J. Softw.*, vol. 26, no. 1, pp. 26–39, 2015.

[80] F. Zhuang, P. Luo, H. Xiong, Q. He, Y. Xiong, and Z. Shi, "Exploiting associations between word clusters and document classes for cross-domain text categorization," *Stat. Anal. Data Mining*, vol. 4, no. 1, pp. 100–114, 2011.



Lutao Zheng received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2011, and the M.S. degree from the School of Computer and Information, Normal of Anhui University, Anhui, China, in 2015.

He is currently working at CaiTong Security Company Ltd., Hangzhou, China. His current research interests include credit card fraud detection, machine learning, deep learning, big data, transfer learning, and natural language processing.

Guanjun Liu (Senior Member, IEEE) received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2011.

He was a Post-Doctoral Research Fellow with the Singapore University of Technology and Design, Singapore, from 2011 to 2013. He was a Post-Doctoral Research Fellow with the Humboldt University of Berlin, Berlin, Germany, from 2013 to 2014, supported by the Alexander von Humboldt Foundation. He is currently a Professor with the Department of Computer Science and Technology, Tongji University. He has authored over 100 articles, including 21 in IEEE/ACM Transactions and 2 books. His research interests include Petri net theory, model checking, machine learning, and information security.

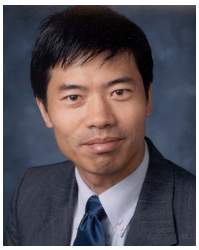
Chungang Yan received the Ph.D. degree from Tongji University, Shanghai, China, in 2006.

She is currently a Professor with the Department of Computer Science and Technology, Tongji University. She has published more than 100 articles in domestic and international academic journals and conference proceedings. Her current research interests include concurrent model and algorithm, Petri net theory, formal verification of software, and trusty theory on the software process.

Changjun Jiang received the Ph.D. degree from the Institute of Automation, Chinese Academy of Science, Beijing, China, in 1995.

He is currently the Leader of the Key Laboratory of the Ministry of Education for Embedded System and Service Computing, Tongji University, Shanghai, China. He is an Honorary Professor with Brunel University London, Uxbridge, U.K. He has published more than 300 articles in journals and conference proceedings. He has led over 30 projects. His research interests include concurrence theory, Petri nets, formal verification of software, cluster, grid technology, intelligent transportation systems, and service-oriented computing.

Dr. Jiang is a fellow of IET. He was a recipient of one international prize and seven prizes in the field of science and technology.



Mengchu Zhou (Fellow, IEEE) joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and is currently a Distinguished Professor. He has over 800 publications, including 12 books, over 500 journal articles (over 400 in the IEEE transactions), 26 patents, and 29 book chapters. His interests are in Petri nets, intelligent automation, the Internet of Things, and big data.

Dr. Zhou is a fellow of the IFAC, the AAAS, and the CAA. He was a recipient of the Humboldt Research Award for U.S. Senior Scientists from the Alexander von Humboldt Foundation, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society. He is the founding Editor of the IEEE Press Book Series on Systems Science and Engineering and the Editor-in-Chief of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA.

Maozhen Li (Member, IEEE) received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in 1997.

He was a Post-Doctoral Research Fellow with the School of Computer Science and Informatics, Cardiff University, Cardiff, U.K., from 1999 to 2002. He is currently a Professor with the Department of Electronic and Computer Engineering, Brunel University London, Uxbridge, U.K., and also a Visiting Professor with the Department of Computer Science, Tongji University, Shanghai, China. His research interests include the areas of high-performance computing, big data analysis, and intelligent systems. He has more than 150 research publications in these areas.

Dr. Li is a fellow of the British Computer Society and the Institute of Engineering and Technology. He is on the editorial boards of a number of journals.