Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 282747, 8 pages
http://dx.doi.org/10.1155/2014/282747

*Research Article*

# Instance Transfer Learning with Multisource Dynamic TrAdaBoost

**Qian Zhang, Haigang Li, Yong Zhang, and Ming Li**

*School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221116, China*

Correspondence should be addressed to Qian Zhang; zhangqian374@126.com

Received 8 April 2014; Revised 5 July 2014; Accepted 11 July 2014; Published 24 July 2014

Academic Editor: Juan R. Rabuñal



Since the transfer learning can employ knowledge in relative domains to help the learning tasks in current target domain, compared with the traditional learning it shows the advantages of reducing the learning cost and improving the learning efficiency. Focused on the situation that sample data from the transfer source domain and the target domain have similar distribution, an instance transfer learning method based on multisource dynamic TrAdaBoost is proposed in this paper. In this method, knowledge from multiple source domains is used well to avoid negative transfer; furthermore, the information that is conducive to target task learning is obtained to train candidate classifiers. The theoretical analysis suggests that the proposed algorithm improves the capability that weight entropy drifts from source to target instances by means of adding the dynamic factor, and the classification effectiveness is better than single source transfer. Finally, experimental results show that the proposed algorithm has higher classification accuracy.

## 1. Introduction

In data mining, a general assumption for the traditional machine learning is that training data and test data have the same distribution. However, in the practical application, this assumption cannot be often met [1]. By transferring and sharing different field knowledge for target task learning, transfer learning makes the traditional learning from scratch an addable one. This must improve the learning efficiency and reduce the learning cost [2, 3]. In 2005, Information Processing Techniques Office (IPTO) gave a new mission of transfer learning: the ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks. In this definition, transfer learning aims to extract the knowledge from one or more source tasks and apply the knowledge to a target task [2]. Since the transfer learning needs to use information from similar domains and tasks, its effectiveness is related to the correlation between the source and target domains.

However, transfer learning is more complex than traditional machine learning because of the introduction of transfer. There are many kinds of knowledge representation in related domains, such as sample instances, feature mapping, model parameters, and association rules. Due to the simpleness of implement, the paper selects sample instances as knowledge representation to design the effective transfer algorithm. In detail, instance transfer learning is used to improve the classification accuracy by finding training samples in other source domains which have strong correlation with the target domain and reusing them in the learning of target task [4]. Obviously, how to decide weight of this training data should influence the effectiveness of candidate classifiers [5].

Up to now, researchers have proposed several approaches to solve transfer learning problems. Ben and Schuller provided a theoretical justification for multitask learning [6]. Daumé and Marcu studied the domain-transfer problem in statistical natural language processing by using a specific Gaussian model [7]. Wu and Dietterich proposed an image classification algorithm by using both inadequate training data and plenty of low quality auxiliary data [8]. This algorithm demonstrates some improvement by using the auxiliary data, but it does not give a quantitative study using different auxiliary examples. Liao et al. proposed a new active

learning method to select the unlabeled data in a target domain to be labeled with the help of the source domain data [9]. Rosenstein et al. proposed a hierarchical Naive Bayes approach for transfer learning by using auxiliary data and discussed the applying time problem of transfer learning [10].

Transfer AdaBoost algorithm, also called TrAdaBoost, is a classic transfer learning algorithm which is proposed by Dai et al. [11]. TrAdaBoost assumes that the source and target domain data use exactly the same set of features and labels, but the distributions of the data in the two domains are different. In addition, TrAdaBoost assumes that, due to the difference in distributions between the source and the target domains, some of the source domain data may be useful in learning for the target domain but some of them may not and could even be harmful. Since TrAdaBoost relies only on one source, its learning effects will become poor when there is a weak correlation between the source and target domains. Moreover, as the literatures [12–14] said, TrAdaBoost has the weaknesses of weight mismatch, introducing imbalance and rapid convergence of source weights. The purpose of this paper is to remove the weight drift phenomenon efficiently, improve learning efficiency, and inhibit the negative transfer.

## 2. Multisource Dynamic TrAdaBoost Algorithm

Considering the correlation between multiple source domains and the target domain, recently Yao and Doretto proposed multisource TrAdaBoost (MSTrA) transfer learning algorithms [15]. As an instance-based transfer learning method, MSTrA selects its training samples from different source domains. At each iteration, MSTrA always selects the most related source domain to train the weak classifier. Although this can ensure that the knowledge transferred is relevant to the target task, MSTrA ignores effects of other source domains. Samir and Chandan proposed an algorithm (DTrAdaBoost) with an integrated dynamic cost to resolve a major issue in the boosting-based transfer algorithm, TrAdaBoost [16]. This issue causes source instances to converge before they can be used for transfer learning. But DTrAdaBoost has low efficiency of learning.

In order to overcome the above disadvantage, a multisource dynamic TrAdaBoost algorithm (MSDTrA) is proposed. By using this algorithm, the rate of convergence of source sample weight will be reduced based on weak correlation to target domain [17]. Supposing there are $N$ source domains, $D_{S_1}, \ldots, D_{S_N}$; $N$ source tasks, $T_{a_1}, \ldots, T_{a_N}$; and $N$ source training data, $D_{a_1}, \ldots, D_{a_N}$, the purpose of transfer learning is to make good use of them to improve the learning effectiveness of the target classifier function $\widehat{f}_b : X \rightarrow Y$. In detail, the algorithm steps of MSDTrA are described as follows.

*Step 1.* Initialize the weight vector $(\omega_{a_1}, \ldots, \omega_{a_N}, \omega_b)$, where $\omega_{a_k} = (\omega_{a_k}^1, \ldots, \omega_{a_k}^{n_{a_k}})$ are the weight vectors of training samples with $k$th source domain and $\omega_b = (\omega_b^1, \ldots, \omega_b^{n_b})$ are the weight vectors of training samples in target domain.

*Step 2.* Set the value of $\beta_a$ as follows:

$$\beta_a = \frac{1}{1 + \sqrt{2 * \ln(n_a)/M}}, \tag{1}$$

where $n_a = \sum_k n_{a_k}$ is the number of all source domains training samples and $n_{a_k}$ is the sample number of training sets with $k$th source domain.

*Step 3.* Empty the set of candidate weak classifiers and normalize the weight vectors $(\omega_{a_1}, \ldots, \omega_{a_N}, \omega_b)$ to 1.

*Step 4.* Select a base learner to obtain the candidate weak classifiers $(f_b^t)^k$ based on training set $D_{a_k} \cup D_b$; calculate the error of $(f_b^t)^k$ on $D_b$ according to the following equation:

$$\left(\varepsilon_b^t\right)^k = \sum_{j=1}^{n_b} \frac{\omega_b^j \sum_{k=1}^{N} \left[ y_b^j \neq \left(f_b^t\right)^k \cdot x_b^j \right]}{\sum_{i=1}^{n_b} \omega_b^i}; \tag{2}$$

update the weight of $(f_b^t)^k$ by using the vectors update strategy:

$$\left(\omega_b^t\right)^k = \frac{e^{1-\left(\varepsilon_b^t\right)^k}}{e^{\left(\varepsilon_b^t\right)^k}}. \tag{3}$$

Repeat the above method until all source domains are traversed, where $(\varepsilon_b^t)^k$ is the error rate of candidate weak classifiers with $k$th source domains in target domain. $y_b^j \neq (f_b^t)^k \cdot x_b^j$ stands for error classified with the candidate weak classifiers. According to the vectors update strategy above, the error of each weak classifier in the target training set is computed and a weight is assigned to each weak classifier according to the error. The larger the error is, the smaller the weight becomes. In other words, source domains which correspond to those classifiers with high classification accuracy contain much valuable information for the learning of target task.

*Step 5.* Integrate all weighted weak classifiers to obtain a candidate classifier at the $t$th iteration:

$$f_b^t = \sum_k \frac{\left(\omega_b^t\right)^k}{\sum_k \left(\omega_b^t\right)^k} \left(f_b^t\right)^k, \tag{4}$$

where the classification error of $f_b^t$ on $D_b$ at iteration $t$ is

$$\varepsilon_b^t = \sum_{j=1}^{n_b} \frac{\omega_b^j \sum_{t=1}^{M} \left[ y_b^j \neq f_b^t \cdot x_b^j \right]}{\sum_{i=1}^{n_b} \omega_b^i}, \tag{5}$$

where $\varepsilon_b^t$ must be less than 0.5. Then, calculate the errors of the candidate classifier on the source and target training sets, based on which update the weights of training samples on the source and target domains. For the correct classified source training samples, their corresponding weights keep unchanged.

*Step 6.* Set

$$\beta_b^t = \frac{\varepsilon_b^t}{1-\varepsilon_b^t}, \quad C_t = 2\left(1-\varepsilon_b^t\right), \quad 0 \le \varepsilon_b^t \le \frac{1}{2}, \tag{6}$$

where $C_t = 2(1-\varepsilon_b^t)$ is the expression of dynamic factor $C_t$. And Theorem 1 will provide the deduce process.

*Step 7.* Update the weight vector of source samples according to the following rule:

$$\omega_{a_k}^{(t+1)\cdot i} = C_t \cdot \omega_{a_k}^{t\cdot i} \cdot (\beta_a)^{\sum_{t=1}^{M}[y_b^j \neq f_b^t \cdot x_b^j]}, \quad i \in D_{a_k}. \tag{7}$$

Update the weight of target samples according to the rule:

$$\omega_b^{(t+1)\cdot i} = \omega_b^{t\cdot i} \cdot \left(\beta_b^t\right)^{\sum_{t=1}^{M}[y_b^j \neq f_b^t \cdot x_b^j]}, \quad i \in D_b, \tag{8}$$

where the weight update of the source instances uses the weighted majority algorithm (WMA) mechanism. This updated mechanism is computed by $\beta_a$ and $C_t$. The target instance weights are updated by using $\varepsilon_b^t$, which is calculated on Step 6.

*Step 8.* Retrain all weak classifiers using the training samples with updated weights. If the maximum number of iterations is reached, $t < M$, return to Step 3; otherwise, turn to Step 9.

*Step 9.* Decide the final strong classifier

$$\widehat{f}_b = \operatorname{sign}\left\{\prod_{t=1}^{M}\left[\left(\beta_b^t\right)^{-f_b^t}\right] - \prod_{t=1}^{M}\left[\left(\beta_b^t\right)^{-1/2}\right]\right\}. \tag{9}$$

In the MSDTrA algorithm, TrAdaBoost's ensemble learning is selected to train classifiers based on the combination set of source and target instances in every step. WMA is used to adjust weights of the source set by decreasing the weight of misclassified source instances and preserving current weights of correctly classified source instances.

It can be seen from the above algorithm that the MSDTrA allows all source training samples to participate in learning process at each iteration, and different source training samples are assigned different weights. If a source training sample can improve the learning of target task, it will be assigned a large weight. Overall, the MSDTrA takes full advantage of all useful knowledge from all source domains, and this can obviously enhance the learning effectiveness of target task.

## 3. Theoretical Analysis

The previous section introduced in detail the proposed new algorithm, that is, the instance transfer learning algorithm. In this section, related theory analyses will be given according to single source TrAdaBoost algorithm [13]. First, Theorems 1 and 2 will proof the influence of source and target sample weight vectors with dynamic factor in source weight, respectively.

**Theorem 1.** *A dynamic factor of $C_t = 2(1-\varepsilon_b^t)$ that is applied to the source weights can prevent their weight drift and get the weight vector to update mechanism of source sample.*

*Proof.* Set $A$ is sum of correctly classified target weights at boosting iteration $t+1$ and $B$ is sum of misclassified target weights at boosting iteration $t+1$. Consider

$$A = n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right) \cdot \left(\frac{\varepsilon_b^t}{1-\varepsilon_b^t}\right)^{\sum_{t=1}^{M}[y_b^j = f_b^t \cdot x_b^j]} = n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right)$$
$$\text{if } \sum_{t=1}^{M}\left[y_b^j = f_b^t \cdot x_b^j\right] = 0,$$
$$B = n_b \cdot \omega_b^t \cdot \varepsilon_b^t \cdot \left(\frac{\varepsilon_b^t}{1-\varepsilon_b^t}\right)^{\sum_{t=1}^{M}[y_b^j = f_b^t \cdot x_b^j]} = n_a \cdot \omega_{a_k}^t\left(1-\varepsilon_b^t\right)$$
$$\text{if } \sum_{t=1}^{M}\left[y_b^j \neq f_b^t \cdot x_b^j\right] = 1. \tag{10}$$

Substituting for $A$ and $B$ to simplify the source update of TrAdaBoost, we have

$$\omega_{a_k}^{t+1} = \frac{\omega_{a_k}^t}{n_a \cdot \omega_{a_k}^t + A + B} = \frac{\omega_{a_k}^t}{n_a \cdot \omega_{a_k}^t + 2 \cdot n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right)}. \tag{11}$$

Introducing the correction factor into the WMA, because of $\omega_{a_k}^{t+1} = \omega_{a_k}^t$, we have

$$\omega_{a_k}^t = \frac{C_t \cdot \omega_{a_k}^t}{C_t \cdot n_a \cdot \omega_{a_k}^t + 2 \cdot n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right)},$$
$$C_t = \frac{2 \cdot n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right)}{1 - n_a \cdot \omega_{a_k}^t} = \frac{2 \cdot n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right)}{n_b \cdot \omega_b^t} = 2\left(1-\varepsilon_b^t\right). \tag{12}$$

□

**Theorem 2.** *The dynamic factor of $C_t = 2(1-\varepsilon_b^t)$ that is applied to the source weights makes the target weights converge as outlined by TrAdaBoost.*

*Proof.* In TrAdaBoost, without any source instances ($n_a = 0$), target weights for correctly classified instances will be updated as

$$\omega_b^{t+1} = \frac{\omega_b^t}{\sum_{j=1}^{n_b}\omega_b^t \cdot \left(\varepsilon_b^t/(1-\varepsilon_b^t)\right)^{\sum_{t=1}^{M}[y_b^j \neq f_b^t \cdot x_b^j]}} = \frac{\omega_b^t}{A+B}$$
$$= \frac{\omega_b^t}{2 \cdot n_b \cdot \omega_b^t\left(1-\varepsilon_b^t\right)} = \frac{\omega_b^t}{2\,(1)\left(1-\varepsilon_b^t\right)}. \tag{13}$$

Applying the dynamic factor to update the source instance weight, we can get the update mechanism of the target instance weight based on MSDTrA. Consider

$$\begin{aligned}\omega_b^{t+1} &= \frac{\omega_b^t}{n_a \cdot \omega_{a_k}^t + 2n_b \cdot \omega_b^t \left(1-\varepsilon_b^t\right)} \\ &= \frac{\omega_b^t}{C_t \cdot n_a \cdot \omega_{a_k}^t + 2n_b \cdot \omega_b^t \left(1-\varepsilon_b^t\right)} \\ &= \frac{\omega_b^t}{2\left(1-\varepsilon_b^t\right) \cdot n_a \cdot \omega_{a_k}^t + 2n_b \cdot \omega_b^t \left(1-\varepsilon_b^t\right)} \\ &= \frac{\omega_b^t}{2\left(1-\varepsilon_b^t\right)\left(n_a \cdot \omega_{a_k}^t + n_b \cdot \omega_b^t\right)} \\ &= \frac{\omega_b^t}{2\left(1-\varepsilon_b^t\right)(1)}.\end{aligned} \tag{14}$$

□

Next, we analysis the performance of MSDTrA on the target training set.

**Theorem 3.** *The final error on the target training set is*

$$\varepsilon \le 2^M \cdot \prod_{t=1}^{M} \sqrt{\varepsilon_b^t\left(1-\varepsilon_b^t\right)}. \tag{15}$$

*Proof.* Supposing that the final sample set which contains all misclassified samples on the target domain is $T$, the final error is $\varepsilon = |T|/n_b$.

At each iteration, the error on the target training set is

$$\varepsilon_b^t = \sum_k \frac{\left(w_b^t\right)^k}{\sum_k \left(w_b^t\right)^k}\left(\varepsilon_b^t\right)^k = \frac{\sum_k e^{1-2\left(\varepsilon_b^t\right)^k}\left(\varepsilon_b^t\right)^k}{\sum_k e^{1-2\left(\varepsilon_b^t\right)^k}}, \tag{16}$$

where $0 \le \left(\varepsilon_b^t\right)^k \le 1/2$.

If the error on the target training set is 0, $\varepsilon_b^t = 0$, training sample weights are not updated, $w_b^{(t+1)i} = w_b^{ti}$. If $\varepsilon_b^t \ne 0$ and $\beta_b^t = \varepsilon_b^t/(1-\varepsilon_b^t) \ne 0$, the updating rule for the weights of target training samples is as follows:

$$\begin{aligned}\sum_{i=1}^{n_b} w_b^{(t+1)i} &= \sum_{i=1}^{n_b} w_b^{t\cdot i}\left(\beta_b^t\right)^{1-\varepsilon_b^t} \\ &\le \sum_{i=1}^{n_b} w_b^{t\cdot i}\left(1-\left(1-\varepsilon_b^t\right)\beta_b^t\right).\end{aligned} \tag{17}$$

Then,

$$\begin{aligned}\sum_{i=1}^{n_b} w_b^{(M+1)i} &= \sum_{i=1}^{n_b} w_b^{t\cdot i}\left(\beta_b^t\right)^{1-\varepsilon_b^t} \\ &\le \sum_{i=1}^{n_b} w_b^i \prod_{t=1}^{M}\left(1-\left(1-\varepsilon_b^t\right)\beta_b^t\right) \\ &:= \prod_{t=1}^{M}\left(1-\left(1-\varepsilon_b^t\right)\beta_b^t\right).\end{aligned} \tag{18}$$

In addition, we have the following criterion:

$$\sum_{i=1}^{n_b} w_b^{(M+1)i} \ge \sum_{i\in S} w_b^i\left(\prod_{t=1}^{M}\beta_b^t\right)^{1/2} = \varepsilon\left(\prod_{t=1}^{M}\beta_b^t\right)^{1/2}. \tag{19}$$

Combining (18) and (19), we have

$$\varepsilon\left(\prod_{t=1}^{M}\beta_b^t\right)^{1/2} \le \prod_{t=1}^{M}\left(1-\left(1-\varepsilon_b^t\right)\beta_b^t\right). \tag{20}$$

Substituting $\beta_b^t = \varepsilon_b^t/(1-\varepsilon_b^t)$ into (20), we can obtain

$$\varepsilon \le 2^M \cdot \prod_{t=1}^{M}\sqrt{\varepsilon_b^t\left(1-\varepsilon_b^t\right)}. \tag{21}$$

□

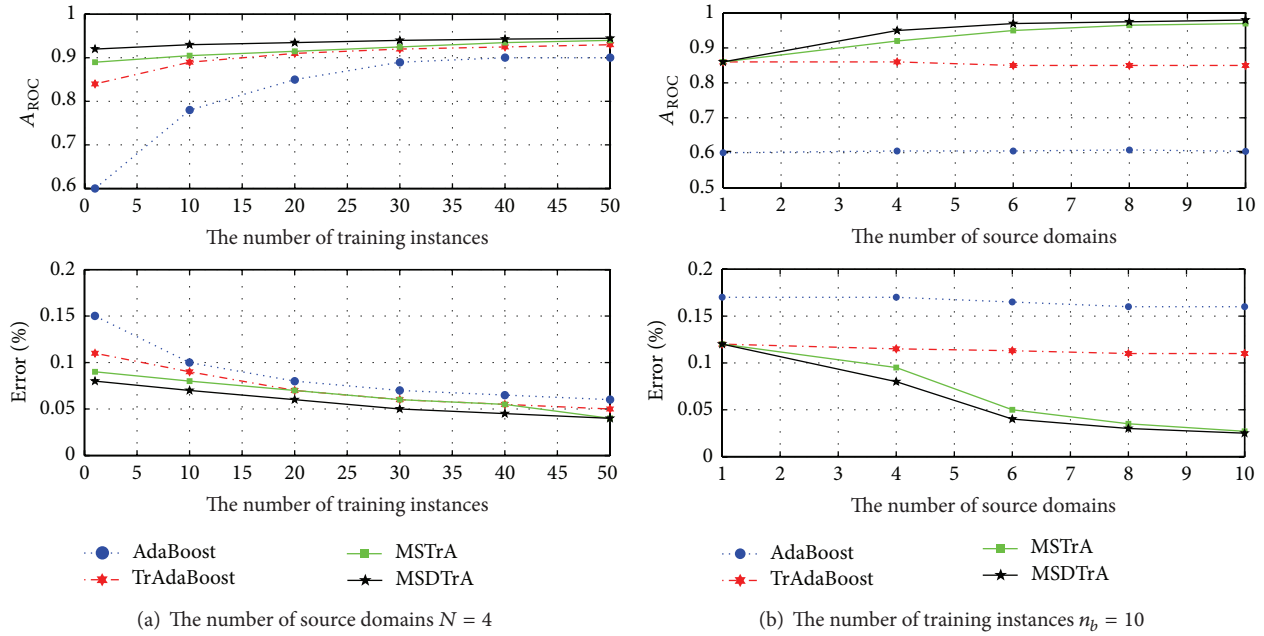
According to Theorem 3, because the condition of $\varepsilon_b^t < 0.5$ is satisfied in the algorithm, the error in final target training data will decrease with the increase of iterations. And the upper bound of the associated generalization error can be calculated by $\varepsilon + O(\sqrt{Md_{\mathrm{VC}}/n_b})$, where $d_{\mathrm{VC}}$ is the VC-dimension of the weak classifier model.

## 4. Experimental Results and Analysis

The performance of the proposed method is investigated based on object category recognition in this section. Without loss of generality, we consider the following case: a small number of training samples of a target object category and a large number of training samples of other source object categories. For any test sample, we verify whether it belongs to the target object category or not.

*4.1. Experimental Setting.* For object category recognition, the Caltech 256 datasets that contain 256 object categories are considered. Practically, among 256 object categories, the 80 categories that contain more than 50 samples are used in our experiment. We designate the target category and randomly draw the samples that form the target data. The number of samples for training $n_b$ is limited between 1 and 50, while the number of samples for testing is 50. Furthermore, in order to illustrate the proposed method does not depend on the data set, we have also used the background dataset, collected via the Google image search engine, along with the remaining categories as our augmented background data set, to verify the effectiveness and robustness of this method.

The remaining categories are treated as the repository from which to draw positive samples for the source data. The numbers of source categories or domains are varied from 1 to 10 in order to investigate the performance of the classifiers with respect to the variability of domains. The number of samples for one source of data is 100. For each target object category, the performance of the classifier is evaluated over 20 random combinations of $N$ source object categories. Given the target and source categories, the performance of the classifier is obtained by averaging over 20 trials of experiments. The overall performance of the classifier is averaged over 20 target categories. SVM is selected as base classifiers and the iteration is 50.




(a) The number of source domains $N = 4$

(b) The number of training instances $n_b = 10$

Figure 1: Performance comparison.

*4.2. Error Analysis.* Since transfer learning is not needed to get good classification results when the target data set is large, standard cross-validation method is not used here. Small portion data of the target set are used for training, and most of the remaining samples are used for testing. Figure 1 compares AdaBoost, TrAdaBoost, MSTrA, and MSDTrA based on the area under the receiver operating characteristic curve (ROC) with different number of target training samples ($n_b = \{1, 10, 20, 30, 40, 50\}$) and different number of sources in the field ($N = \{1, 4, 6, 8, 10\}$). Moreover, for the area bounded by the ROC curve and the $X$-axis, $A_{\text{ROC}}$ is used to evaluate the performance of different algorithms.

Practically, fixing the number of source domains $N = 4$, Figure 1(a) shows the ROC curves of the four algorithms with the increase of the number of training instances. Since AdaBoost does not transfer any knowledge from the source, its performance depends mainly on the number of $n_b$. For a very small value of $n_b$, it performs slightly improvement as the ROC curves show. However, due to the transfer learning mechanism, TrAdaBoost has good improvement by combining the three sources. By incorporating the ability to transfer knowledge from multiple individual domains, MSTrA and MSDTrA demonstrate a significant improvement in recognition accuracy, even for a very small $n_b$. In addition, the performance of AdaBoost and TrAdaBoost strongly depends on the selection of source domains and target positive samples, as the standard deviation of $A_{\text{ROC}}$ shows.

Fixing the number of training instances $n_b = 10$, Figure 1(b) shows the ROC curves of the four algorithms with increase of the number of source domains. We can see that as the number of source domains increase, the $A_{\text{ROC}}$ of MSTrA and MSDTrA increases and the corresponding standard deviations also decrease. This indicates an improved performance in both accuracy and consistency. Since TrAdaBoost is incapable of exploring the decision boundaries separating multiple source domains, its performance keeps unchanged regardless of the number of source domains.

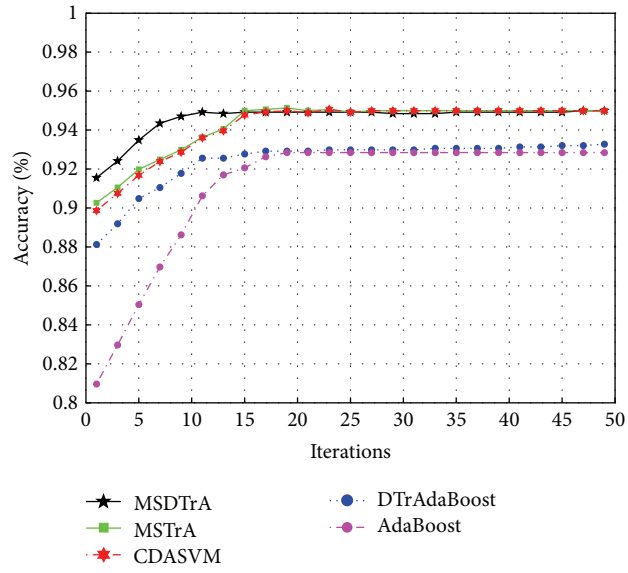
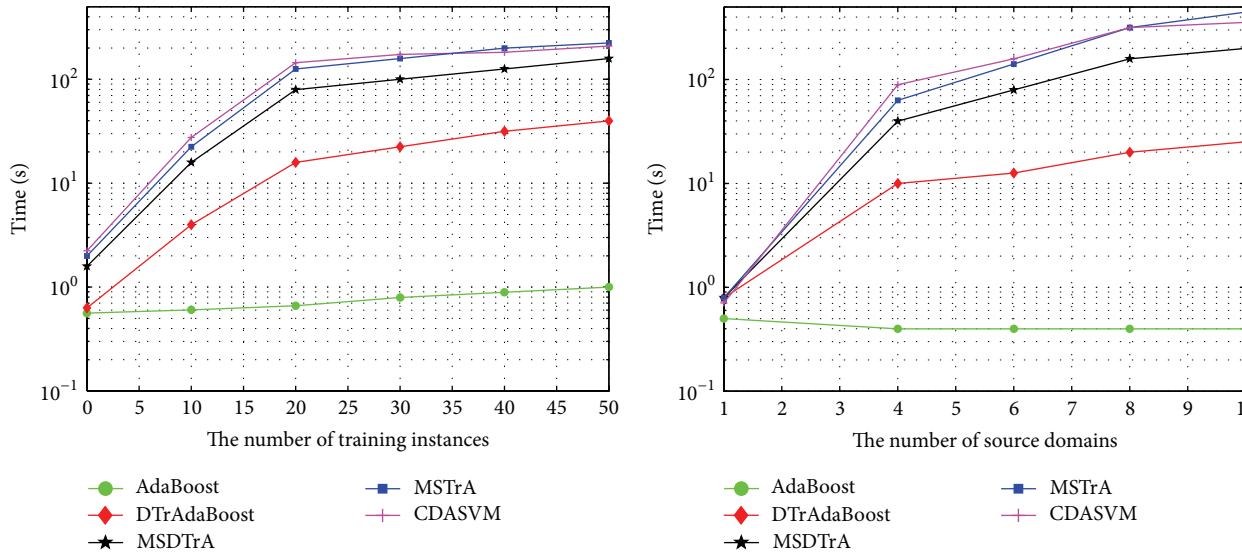



Figure 2: The classification performance on the target domain.

Figure 2 compares the classification performance of different methods in the target domain. We can see that AdaBoost algorithm does not transfer source domain knowledge and gets lower classification accuracy. DTrAdaBoost has relatively poor test results, because it only uses one source




(a) The number of source domains $N = 4$

(b) The number of training instances $n_b = 10$

FIGURE 3: Time cost comparison on different methods.

domain training sample set and gains the least useful knowledge from source domain. CDASVM based on structural risk minimization model fully considers the source domain sample information, and thus it has good classification accuracy. MSTrA and MSDTrA use four different sources domain training sets which contain more useful information, so they get higher testing accuracy than other algorithms. In each set of experiments, MSTrA only selects classification with the highest accuracy at each iteration and ignores the impact of other source domains to target tasks. But by adding the dynamic factors and weighting mechanism, MSDTrA makes better use of all sources domains useful knowledge and eliminates the influences of unrelated samples in sources domains training set to target tasks, so it has better performance than MSTrA algorithm.

In order to have objective and scientific comparison results, hypothesis testing is used on the experimental results. Let the variables $X_1, X_2, X_3, X_4, X_5$ denote the classification error rate of MSDTrA, MSTrA, CDASVM, DTrAdaBoost, and AdaBoost algorithms, respectively. Since the value of $X_1$, $X_2, X_3, X_4, X_5$ is subject to many random factors, we assume that they submit to normal distribution, $X_i \sim N(\mu_i, \sigma_i^2)$, $i = 1, 2, 3, 4, 5$. Now, we compare the random variable means of these algorithms, $\mu_i$ $(i = 1, 2, 3, 4, 5)$. The smaller the $\mu_i$ is, the lower the expected classification error rate is and the higher the efficiency is. Because the sample variance is the unbiased estimation of the overall variance, the sample variance value is used as an estimate of the generality variance. In this experiment the significance level $\alpha$ is set as 0.01.

Table 1 shows the comparison process on $\mu_i$ and other parameters. We can see from Table 1 that the expectations of classification error rate in MSDTrA is far below than other algorithms.

*4.3. Time Complexity.* Since several domains are used into the learning of target task together, time complexity of multisource domains is more than single domain. Supposing that the time complexities of training a classifier and updating weight are $C_h$ and $C_w$, respectively, the time complexity of AdaBoos, DTrAdaBoost, MSTrA, and MSDTrA can be approximated to $C_hO(M) + C_wO(n_bM)$, $C_hO(M) + C_wO(n_aM)$, $C_hO(NM) + C_wO(n_aNM)$ and $C_hO(NM) + C_wO(n_aM)$. Furthermore, Figure 3 shows the average training time of the four algorithms with fixed $N$, $n_b$.

*4.4. Dynamic Factor.* This experiment will prove the effect of dynamic factor on source weights and target weights. Here a sources domain is considered, $N = 1$. In Figure 4(a), the number of instances is set as constant ($n_a = 1000$, $n_b = 200$) and the source error rate is set to zero. According to the WMA, the weights should not change because of $\varepsilon_{a_k}^t = 0$; that is, $\omega_{a_k}^{t+1} = \omega_{a_k}^t$. When target error rates $\varepsilon_b^t = \{10\%, 20\%, 30\%, 40\%\}$, the ratio of the weights of MSDTrA and MSTrA is plotted at different boosting iterations.

We can see from Figure 4(a) the following. (1) In MSTrA, source weights converge always even the classification results are correct. (2) MSDTrA matches the behavior of the WMA. (3) If dynamic factor is not applied, the smaller the value of $\varepsilon_b^t$ is and the faster the convergence rate of source weights is. In addition, for a weak learner with $\varepsilon_b^t = 10\%$, MSTrA is still not able to get good performance by using over 1000 source instances, even though they were never misclassified.

*4.5. Rate of Convergence.* The number of source instances was set ($n_b = 1000$), and the classification error is permitted to vary within the range of $\varepsilon_b^t \in \{10\% \sim 50\%\}$; Figure 4(b) shows results after a single iteration with different number

Table 1: Hypothesis testing for experimental results.

| Hypothesis | $H_0: \mu_1 \geq \mu_2$<br>$H_1: \mu_1 < \mu_2$ | $H_0: \mu_1 \geq \mu_3$<br>$H_1: \mu_1 < \mu_3$ | $H_0: \mu_1 \geq \mu_4$<br>$H_1: \mu_1 < \mu_4$ | $H_0: \mu_1 \geq \mu_5$<br>$H_1: \mu_1 < \mu_5$ |
|---|---|---|---|---|
| Statistics | $U_1 = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\sigma_1^2/n_1 + \sigma_2^2/n_2}}$ | $U_2 = \frac{\overline{X}_1 - \overline{X}_3}{\sqrt{\sigma_1^2/n_1 + \sigma_3^2/n_3}}$ | $U_3 = \frac{\overline{X}_1 - \overline{X}_4}{\sqrt{\sigma_1^2/n_1 + \sigma_4^2/n_4}}$ | $U_4 = \frac{\overline{X}_1 - \overline{X}_5}{\sqrt{\sigma_1^2/n_1 + \sigma_5^2/n_5}}$ |
| Rejection region | $U_1 \leq -Z_\alpha = -2.325$ | $U_2 \leq -Z_\alpha = -2.325$ | $U_3 \leq -Z_\alpha = -2.325$ | $U_4 \leq -Z_\alpha = -2.325$ |
| Value of the statistic | $U_1 = -58.67$ | $U_2 = -114.56$ | $U_3 = -136.59$ | $U_4 = -158.23$ |
| Conclusion | $H_1: \mu_1 < \mu_2$ | $H_1: \mu_1 < \mu_3$ | $H_1: \mu_1 < \mu_4$ | $H_1: \mu_1 < \mu_5$ |

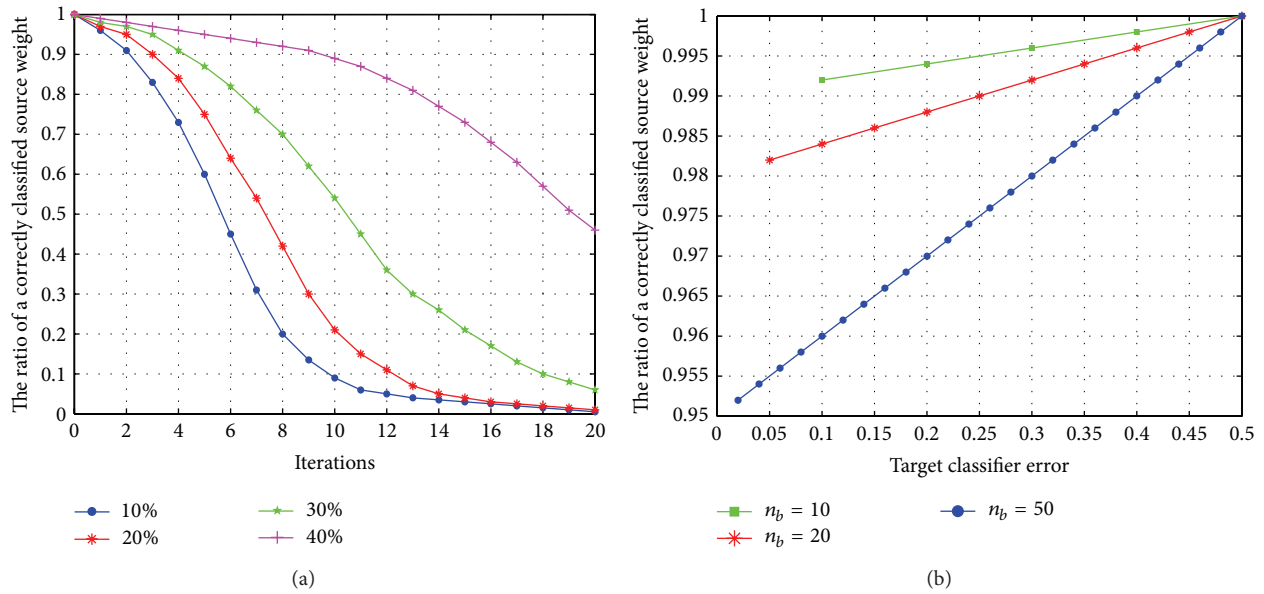



Figure 4: (a) Results for 20 iterations with different target error rates. (b) Results after a single iteration with different number of target instances.

of target instances $\{10, 20, 50\}$. It can be observed that after a single boosting iteration, the ratio of a correctly classified source instances increases with the increases of $\varepsilon_b^t$.

## 5. Conclusions

Considering the situation that sample data from the transfer source domain and the target domain have similar distribution, an instance transfer learning method based on multisource dynamic TrAdaBoost is provided. By integrating with the knowledge in multiple source domains, this method makes good use of the information of all source domains to guide the target task learning. Whenever candidate classifiers are trained, all the samples in all source domains are involved in learning, and the information that is beneficial to target task learning can be obtained, so that negative transfer can be avoided. The theoretical analysis and experimental results suggest that the proposed algorithm has higher classification accuracy compared with several existing algorithms.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgment

This research is supported by the Fundamental Research Funds for the Central Universities (2013XK09).

## References

[1] H. Wang, Y. Gao, and X. G. Chen, "Transfer of reinforcement learning: the state of the art," *Acta Electronica Sinica*, vol. 36, pp. 39–43, 2008.

[2] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: a survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[3] Q. Zhang, M. Li, and Y. H. Chen, "Instance-based transfer learning method with multi-source dynamic TrAdaBoost,"

*Journal of China University of Mining and Technology*, vol. 43, no. 4, pp. 701–708, 2014.

[4] J. N. Meng, *Research on the Application of Transfer Learning on Text Classification*, Dalian University of Technology, 2011.

[5] Y. Cheng, G. Cao, X. Wang, and J. Pan, "Weighted multi-source TrAdaBoost," *Chinese Journal of Electronics*, vol. 22, no. 3, pp. 505–510, 2013.

[6] D. S. Ben and R. Schuller, "Exploiting task relatedness for multiple task learning," in *Proceedings of the 16th Annual Conference on Learning Theory*, pp. 567–580, Washington, DC, USA, 2008.

[7] I. Daumé and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, vol. 26, pp. 101–126, 2006.

[8] P. Wu and T. G. Dietterich, "Improving SVM accuracy by training on auxiliary data sources," in *Proceedingsof the 21th International Conference on Machine Learning (ICML '04)*, pp. 871–878, July 2004.

[9] X. Liao, Y. Xue, and L. Carin, "Logistic regression with an auxiliary data source," in *Proceedings of the 22nd International Conference on Machine Learning*, pp. 505–512, ACM, August 2005.

[10] M. T. Rosenstein, Z. Marx, L. P. Kaelbling et al., "To transfer or not to transfer," in *Proceedings of the Neural Information Processing Systems Workshop on Transfer Learning (NIPS '05)*, p. 898, 2005.

[11] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 193–200, New York, NY, USA, June 2007.

[12] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, pp. 863–870, Haifa, Israel, June 2010.

[13] E. Eaton and M. Desjardins, "Set-based boosting for instance-level transfer," in *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW '09)*, pp. 422–428, December 2009.

[14] E. Eaton, *Selective knowledge transfer for machine learning [Ph. D. dissertation]*, University of Maryland, Baltimore, Md, USA, 2009.

[15] Y. Yao and G. Doretto, "Boosting for transfer learning with multiple sources," in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 1855–1862, June 2010.

[16] A. S. Samir and K. R. Chandan, "Adaptive boosting for transfer learning using dynamic updates," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 60–75, 2011.

[17] Q. Zhang, M. Li, X. S. Wang et al., "Instance-based transfer learning for multi-source domains," *Acta Automatica Sinica*, vol. 40, no. 6, pp. 1175–1182, 2014.