Boosting for Transfer Learning

Wenyuan Dai DWYAK@APEX.SJTU.EDU.CN
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China

Qiang Yang QYANG@CSE.UST.HK
Deptarment of Computer Science, Hong Kong University of Science and Technology, Hong Kong

Gui-Rong Xue GRXUE@APEX.SJTU.EDU.CN
Yong Yu YYU@APEX.SJTU.EDU.CN
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China

Abstract

Traditional machine learning makes a basic assumption: the training and test data should be under the same distribution. However, in many cases, this *identical-distribution* assumption does not hold. The assumption might be violated when a task from one new domain comes, while there are only labeled data from a similar old domain. Labeling the new data can be costly and it would also be a waste to throw away *all* the old data. In this paper, we present a novel transfer learning framework called `TrAdaBoost`, which extends boosting-based learning algorithms (Freund & Schapire, 1997). `TrAdaBoost` allows users to utilize a small amount of newly labeled data to leverage the old data to construct a high-quality classification model for the new data. We show that this method can allow us to learn an accurate model using only a tiny amount of new data and a large amount of old data, even when the new data are not sufficient to train a model alone. We show that `TrAdaBoost` allows knowledge to be effectively transferred from the old data to the new. The effectiveness of our algorithm is analyzed theoretically and empirically to show that our iterative algorithm can converge well to an accurate model.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

1. Introduction

A fundamental assumption in classification learning is that the data distributions of training and test sets should be identical. When the assumption does not hold, traditional classification methods might perform worse. However, in practice, this assumption may not always hold. For example, in Web mining, the Web data used in training a Web-page classification model can be easily out-dated when applied to the Web sometime later, because the topics on the web change frequently. Often, new data are expensive to label and thus their quantities are limited due to cost issues. How to accurately classify the new test data by making the maximum use of the old data becomes a critical problem.

Although the training data are more or less out-dated, there are certain parts of the data that can still be reused. That is, knowledge learned from this part of the data can still be of use in training a classifier for the new data. To find out which those data are, we employ a small amount of labeled *new* data, called *same-distribution* training data, to help vote on the usefulness of each of the old data instance. Because some of the old training data might be out-dated and be under a different distribution from the new data, we call them *diff-distribution* training data. Our goal is to learn a high-quality classification model using both the same-distribution and diff-distribution data. This learning process corresponds to transferring knowledge learned from the old data to new situations – an instance of *transfer learning*.

In this paper, we try to develop a general framework for transfer learning based on (Freund & Schapire, 1997), and analyze the correctness of the general model using the *Probability Approximately Correct* (PAC) theory. Our key idea is to use boosting to filter out the diff-distribution training data that are very different from the same-distribution data by automatically adjusting the weights of training instances. The remaining diff-distribution data are treated as the additional training data which greatly boost the confidence of the learned model even when the same-distribution training data are scarce. Our experimental results support our conjecture that the boosting-based framework, which we implement in an algorithm known as `TrAdaBoost`, is a high-performance and simple transfer learning algorithm. Our theoretical analysis also confirms that our boosting learning method converges well to the desired model.

The rest of the paper is organized as follows. In Section 2, we discuss the related works. In Section 3, we present our formal definitions for the problem and the framework. We present a theoretical analysis in Section 4 to show the properties of our transfer learning framework. Our experimental results and discussion are shown in Section 5. Section 6 concludes the whole paper and gives some future works.

2. Related Work

Transfer learning is what happens when someone finds it much easier to learn to play chess having already learned to play checkers, or to recognize tables having already learned to recognize chairs; or to learn Spanish having already learned Italian[1]. Recently, transfer learning has been recognized as an important topic in machine learning research. Several researchers have proposed new approaches to solve the problems of transfer learning. Early transfer learning works raised some important issues, such as *learning how to learn* (Schmidhuber, 1994), *learning one more thing* (Thrun & Mitchell, 1995), *multi-task learning* (Caruana, 1997). A related topic is multi-task learning whose objective is to discover the common knowledge in multiple tasks. This common knowledge belongs to almost all the tasks, and is helpful for solving a new task. Ben-David and Schuller (2003) provided a theoretical justification for multi-task learning. In contrast, we address on the single-task learning problem, but the distributions of the training and test data differ from each other. DauméIII and Marcu (2006) have studied the domain-transfer problem in statistical natural language processing, using a specific Gaussian model. In this paper, we try to develop a transfer classification framework under the PAC learning model.

[1]http://www.cs.berkeley.edu/~russell/ebtl/

Our problem setting can also be considered as learning with auxiliary data, where the labeled diff-distribution data are treated as the auxiliary data. In previous works, Wu and Dietterich (2004) proposed an image classification algorithm using both inadequate training data and plenty of low quality auxiliary data. They demonstrated some improvement by using the auxiliary data. However, they did not give a quantitative study using different auxiliary examples. Liao et al. (2005) improved learning with auxiliary data using active learning. Rosenstein et al. (2005) proposed a hierarchical Naive Bayes approach for transfer learning using auxiliary data, and discussed when transfer learning would improve the performance and when decrease.

Another closely related task is learning under *sample selection bias* (Zadrozny, 2004) or *covariate shift* (Shimodaira, 2000), which deals with the case when all the same-distribution data are unlabeled. In the Novel-prize work, Heckman (1979) investigated correcting sample selection bias in econometrics. Bickel and Scheffer (2007) studied the sample selection bias problem in the spam filtering domain. Other researches addressing on correcting sample selection bias include (Dudík et al., 2006; Huang et al., 2007) etc.

3. Transfer Learning through TrAdaBoost

To enable transfer learning, we use part of the labeled training data that have the same distribution as the test data to play a role in building the classification model. We call these training data *same-distribution training data*. The quantity of these same-distribution training data is often inadequate to train a good classifier for the test data. The training data, whose distribution may differ from the test data, perhaps because they are out-dated, are called *diff-distribution training data*. These data are assumed to be abundant, but the classifiers learned from these data cannot classify the test data well due to different data distributions.

More formally, let X_s be the *same-distribution instance space*, X_d be the *diff-distribution instance space*, and $Y = \{0, 1\}$ be the set of category labels. A *concept* is a boolean function c mapping from X to Y, where $X = X_s \cup X_d$. The test data set is denoted by $S = \{(x_i^t)\}$, where $x_i^t \in X_s$ $(i = 1, \ldots, k)$. Here, k is the size of the test set S which is unlabeled. The training data set $T \subseteq \{X \times Y\}$ is partitioned into two labeled sets T_d, and T_s. T_d represents the *diff-distribution training data* that $T_d = \{(x_i^d, c(x_i^d))\}$, where $x_i^d \in X_d$ $(i = 1, \ldots, n)$. T_s represents the *same-distribution training data* that $T_s = \{(x_j^s, c(x_j^s))\}$, where $x_j^s \in X_s$ $(j = 1, \ldots, m)$. n and m are the sizes of T_d and T_s, respectively. $c(x)$ returns the label for the data instance x. The combined training set $T = \{(x_i, c(x_i))\}$ is defined as follows

$$x_i = \begin{cases} x_i^d, & i = 1, \ldots, n; \\ x_i^s, & i = n+1, \ldots, n+m. \end{cases}$$

Here, T_d corresponds to some labeled data from an old domain that we try to reuse as much as we can; however we do not know which part of T_d is useful to us. What we can do is to label a small amount of data from the new domain and call it T_s, and then use these data to find out the useful part of T_d. The *problem* that we are trying to solve is: given a small number of labeled same-distribution training data T_s, many diff-distribution training data T_d and some unlabeled test data S, the objective is to train a classifier $\hat{c} : X \rightarrow Y$ that minimizes the prediction error on the unlabeled data set S.

We now present our *Transfer AdaBoost* learning framework `TrAdaBoost`, which extends `AdaBoost` for transfer learning. `AdaBoost` (Freund & Schapire, 1997) is a learning framework which aims to boost the accuracy of a weak learner by carefully adjusting the weights of training instances and learn a classifier accordingly. However, `AdaBoost` is similar to most traditional machine learning methods by assuming the distributions of the training and test data to be identical. In our extension to `AdaBoost`, `AdaBoost` is still applied to same-distribution training data to build the base of the model. But, for diff-distribution training instances, when they are wrongly predicted due to distribution changes by the learned model, these instances could be those that are the most dissimilar to the same-distribution instances. Thus, in our extension, we add a mechanism *to decrease the weights of these instances in order to weaken their impacts*.

A formal description of the framework is given in Algorithm 1. As can be seen from the algorithm, in each iteration round, if a diff-distribution training instance is mistakenly predicted, the instance may likely conflict with the same-distribution training data. Then, we decrease its training weight to reduce its effect through multiplying its weight by $\beta^{|h_t(x_i)-c(x_i)|}$. Note that $\beta^{|h_t(x_i)-c(x_i)|} \in (0, 1]$. Thus, in the next round, the misclassified diff-distribution training instances, which are dissimilar to the same-distribution ones, will affect the learning process less than the current round. After several iterations, the diff-distribution training instances that fit the same-distribution ones better will have larger training weights, while the diff-distribution training instances that are *dissimilar* to the same-distribution ones will have *lower* weights. The instances with large training weights will intent to help the learning algorithm to train better classifiers.

Algorithm 1 `TrAdaBoost`

```
Input the two labeled data sets $T_d$ and $T_s$, the unlabeled data set $S$, a base learning algorithm Learner, and the maximum number of iterations $N$.
Initialize the initial weight vector, that $\mathbf{w}^1 = (w_1^1, \ldots, w_{n+m}^1)$. We allow the users to specify the initial values for $\mathbf{w}^1$.
For $t = 1, \ldots, N$
    1. Set $\mathbf{p}^t = \mathbf{w}^t / (\sum_{i=1}^{n+m} w_i^t)$.
    2. Call Learner, providing it the combined training set $T$ with the distribution $\mathbf{p}^t$ over $T$ and the unlabeled data set $S$. Then, get back a hypothesis $h_t : X \rightarrow Y$ (or $[0, 1]$ by confidence).
    3. Calculate the error of $h_t$ on $T_s$:
       $\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t \cdot |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}$.
    4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$ and $\beta = 1/(1 + \sqrt{2 \ln n / N})$. Note that, $\epsilon_t$ is required to be less than $1/2$.
    5. Update the new weight vector:
       $w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i)-c(x_i)|}, & 1 \le i \le n \\ w_i^t \beta_t^{-|h_t(x_i)-c(x_i)|}, & n+1 \le i \le n+m \end{cases}$
Output the hypothesis
    $h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & \text{otherwise} \end{cases}$
```

4. Theoretical Analysis of TrAdaBoost

In the last section, we have presented the transfer classification learning framework. In this section, we theoretically analyze our framework in terms of its convergence property, and show why the framework is able to learn knowledge even when the domain distributions are not identical. Our analysis extends the theory of `AdaBoost` (Freund & Schapire, 1997). Due to the limitation of space, we will only give a sketch of the analysis. The details are given in a long version.

Let $l_i^t = |h_t(x_i) - c(x_i)|$ $(i = 1, \ldots, n$ and $t = 1, \ldots, N)$ be the *loss* of the training instance x_i suffered by the hypothesis h_t. The distribution $\mathbf{d}^t$ is the training weight vector with respect to T_d in the t^{th} iteration, that $d_i^t = w_i^t/(\sum_{j=1}^n w_j^t)$, $(i = 1, \dots, n)$. The training loss with respect to T_d suffered by `TrAdaBoost` through N iterations is $L_d = \sum_{t=1}^N \sum_{i=1}^n d_i^t l_i^t$. The training loss with respect to an instance x_i $(i = 1, \dots, n)$ suffered by `TrAdaBoost` through N iterations is $L(x_i) = \sum_{t=1}^N l_i^t$.

In the following, the theoretical conclusions with respect to `TrAdaBoost` are presented. Theorem 1 discusses the convergence property of `TrAdaBoost`. Theorem 2 analyzes the average weighted training loss on diff-distribution data T_d. Theorem 3 shows the prediction error on the same-distribution data T_s. Note that, `TrAdaBoost` optimizes the average weighted training loss on T_d and the prediction error on T_s, simultaneously. Finally, Theorem 4 gives an alternative upper-bound of the generalization error on the same-distribution data.

Theorem 1 *In* `TrAdaBoost`*, when the number of iterations is N, we have*

$$\frac{L_d}{N} \leq \min_{1 \leq i \leq n} \frac{L(x_i)}{N} + \sqrt{\frac{2 \ln n}{N}} + \frac{\ln n}{N}. \qquad (1)$$

Theorem 1 can also be found in (Freund & Schapire, 1997) together with its proof. It indicates that the average training loss (L_d/N) through N iterations on the diff-distribution training data T_d is not much (at most $\sqrt{2 \ln n/N} + \ln n/N$) larger than the average training loss of the instance whose average training loss is minimum (that is, $\min_{1 \leq i \leq n} L(x_i)/N$). From Equation (1), we can also find the convergence rate of `TrAdaBoost` over the diff-distribution training data T_d is $O(\sqrt{\ln n/N})$ in the worst case.

Theorem 2 *In* `TrAdaBoost`*, p_i^t denotes the weight of the training instance x_i, which is defined as $\mathbf{p}^t = \mathbf{w}^t/(\sum_{i=1}^{n+m} w_i^t)$. Then,*

$$\lim_{N \to \infty} \frac{\sum_{t=\lceil N/2 \rceil}^N \sum_{i=1}^n p_i^t l_i^t}{N - \lceil N/2 \rceil} = 0. \qquad (2)$$

Theorem 2 can be derived from Theorem 1. Due to the limitation of space, we omit the proof of Theorem 2 here. Intuitively, Equation (2) indicates that the average weighted training loss suffered by `TrAdaBoost` on the diff-distribution training data T_d from the $\lceil N/2 \rceil^{th}$ iteration to the N^{th} converges to zero. That is the reason we vote the hypothesis h_t from the $\lceil N/2 \rceil^{th}$ iteration to the N^{th} in Algorithm 1. Therefore, `TrAdaBoost` is able to reduce the average weighted training loss on the diff-distribution data.

Theorem 3 below shows that `TrAdaBoost` preserves the similar error-convergence property as the `AdaBoost` algorithm (Freund & Schapire, 1997)

Theorem 3 *Let $I = \{i : h_f(x_i) \neq c(x_i)$ and $n + 1 \leq i \leq n + m\}$. The prediction error on the same-distribution training data T_s suffered by the final hypothesis h_f is defined as $\epsilon = \Pr_{x \in T_s}[h_f(x) \neq c(x)] = |I|/m$. Then,*

$$\epsilon \leq 2^{\lceil N/2 \rceil} \prod_{t=\lceil N/2 \rceil}^{N} \sqrt{\epsilon_t (1 - \epsilon_t)}. \qquad (3)$$

When $\epsilon_t < 0.5$, the prediction error of the final hypothesis h_f on the same-distribution training data T_s becomes increasingly smaller after each iteration.

From Theorems 2 and 3, two conclusions can be reached. First, based on Theorem 3, the final hypothesis h_f produced by `TrAdaBoost` increasingly reduces the error on the same-distribution training data. Second, based on Theorem 2, the weighted average training loss in the diff-distribution part gradually converges to zero. Therefore, `TrAdaBoost` minimizes both the error on the same-distribution training data and the weighted average loss on the diff-distribution training data simultaneously. Intuitively, it can be understood that `TrAdaBoost` first performs its learning on the same-distribution training data, and then chooses the most helpful diff-distribution training data, on condition that they do not result in more average training loss, as additional training data.

We wish to understand why `TrAdaBoost` theoretically improves the implementation of `AdaBoost` with the combined training set T. In fact, `TrAdaBoost` does not guarantee to always improve `AdaBoost`, since the quality of diff-distribution training data is not certain. The goal of `TrAdaBoost`, shown by Theorems 2 and 3, is to reduce the *weighted* training error on the diff-distribution data, while still preserving the properties of `AdaBoost`. The diff-distribution data after reweighting are used as the additional data which might be helpful for training a good prediction model.

All the above analysis focused on the error on the training data and did not address any generalization issues. In Theorem 4, we give an alternative upper-bound of the generalization error on the same-distribution data.

Theorem 4 *Let d_{VC} be the VC-dimension of the hypothesis space, the generalization error on the same-distribution data, with high probability, is at most*

$$\epsilon + O\left(\sqrt{\frac{N d_{\mathrm{VC}}}{m}}\right) \qquad (4)$$

Here, N is the number of iterations, m is the size of the *same-distribution training data T_s, and ϵ is the error on the same-distribution training data from h_f.*

The conclusion given by Theorem 4 is the same as what has been shown in (Schapire, 1999). Theorem 4 presents an upper-bound of the generalization error on the same-distribution data. The upper-bound in Equation (4) shows that the generalization error depends on the same-distribution training error ϵ, the VC-dimension of the hypothesis space d_{VC}, the number of iterations N, and the size m of same-distribution training set. From Equation (4), it can be found that if the size m of the same-distribution training data is small, `TrAdaBoost` may easily overfit when the number of iterations N becomes large. However, as we will show in the experiments, `TrAdaBoost` does not overfit as easily as it is shown theoretically.

5. Experimental Evaluation

5.1. Data Sets

In order to evaluate the properties of our framework, we performed the experiments on three text data sets (20 Newsgroups[2], SRAA[3], Reuters-21578[4]) and one none-text data set (the *mushroom* data set from the UCI machine learning repository[5]). We also split and revised the data sets to fit our transfer learning scenario, to make the distributions of the diff-distribution data T_d and same-distribution data $T_s \cup S$ different.

All three text data sets have hierarchical structures. For example, the 20 Newsgroups corpus contains seven top categories. Under the top categories, there are 20 subcategories. We define the tasks as top-category-classification problems. When we split the data to generate diff-distribution and same-distribution sets, the data are split based on subcategories instead of based on random splitting. Then, the two data sets contain data in different subcategories. Their distributions also differ as a result. For the mushroom data set, since it does not have hierarchy, we split the data set based on the feature *stalk-shape*. The diff-distribution data set consists of all the instances whose stalks are *enlarging*, while the same-distribution data set consists of the instances about*tapering* mushrooms. Then, the two sets contain examples from different types of mushrooms, which makes the distributions different.

Table 1 shows the description for each data set. The first three data sets are from the 20 Newsgroups data set; the next two are from the SRAA corpus; the next three are from the Reuters-21578 text collection; the last data set is generated by the mushroom data set. The name of the data set `rec vs talk` indicates that all the positive instances are from the category `rec`, while negative ones from `talk`. The other data sets are named in the same way. The diff-distribution and same-distribution data sets are split based on subcategories. KL-divergence (Kullback & Leibler, 1951) on the feature space between each corresponding diff-distribution and same-distribution sets is presented in this table. It can be seen that the KL-divergences for all the data sets are much larger than the same-distribution case in which the KL-divergence should be close to zero.

Table 1. The descriptions of the data sets for transfer classification

Data Set	KL-divergence	Size	
		$\lvert T_d\rvert$	$\lvert T_s \cup S\rvert$
rec vs talk	1.102	3,669	3,561
rec vs sci	1.021	3,961	3,965
sci vs talk	0.854	3,374	3,828
auto vs aviation	1.126	8,000	8,000
real vs simulated	1.048	8,000	8,000
orgs vs people	0.303	1,016	1,046
orgs vs places	0.329	1,079	1,080
people vs places	0.307	1,239	1,210
edible vs poisonous	1.315	4,608	3,516

5.2. Comparison Methods

In the experiments, we use Support Vector Machines (Boser et al., 1992; Joachims, 1999) as the basic learners in `TrAdaBoost`. SVM^{light} (Joachims, 2002) with linear kernel is applied in the experiments to implement the `SVM` and `TSVM` classifiers. Furthermore, we also added some constraints to the basic learners to avoid the case of training weights being unbalanced. When training `SVM`, we always balance the overall training weights between positive and negative examples. The constraints designed for `TSVM` (Joachims, 1999) have also been applied to the basic learners. The weights of unlabeled data are set to the default values of SVM^{light}.

Four baseline methods are implemented using SVM^{light} as shown in Table 2. Besides the baselines, we also compare `TrAdaBoost` with the method developed for learning with auxiliary data proposed by Wu and Dietterich (2004), which is denoted as `AUX`. The parameter C^p/C^a (as used in (Wu & Dietterich, 2004)) is set to 4 after tuning.

Our framework `TrAdaBoost` with `SVM` and `TSVM` as the basic learners has been performed in the experiments. We denote them as `TrAdaBoost(SVM)` and

[2]http://people.csail.mit.edu/jrennie/20Newsgroups/
[3]http://www.cs.umass.edu/~mccallum/code-data.html
[4]http://www.daviddlewis.com/resources/testcollections/
[5]http://www.ics.uci.edu/~mlearn/MLRepository.html

Table 2. The descriptions of baseline methods

Baseline	Training Data		Test Data	Basic Learner
	labeled	unlabeled		
`SVM`	T_s	$\emptyset$	S	SVM
`SVMt`	$T_s \cup T_d$	$\emptyset$	S	SVM
`TSVM`	T_s	S	S	TSVM
`TSVMt`	$T_s \cup T_d$	S	S	TSVM

Table 3. The error rates when supervised learning

Data Set	`SVM`	`SVMt`	`AUX`	`TrAdaBoost(SVM)`
rec vs talk	0.222	0.127	0.127	**0.080**
rec vs sci	0.240	0.164	0.153	**0.097**
sci vs talk	0.234	0.177	0.173	**0.125**
auto vs aviation	0.131	0.192	0.188	**0.096**
real vs simulated	0.140	0.219	0.210	**0.119**
orgs vs people	0.494	0.285	0.287	**0.280**
orgs vs places	0.423	0.440	0.433	**0.315**
people vs places	0.412	0.255	0.257	**0.216**
edible vs poisonous	0.127	0.135	0.082	**0.071**

`TrAdaBoost(TSVM)`. In the following, we will use `SVM`, `TSVM`, `TrAdaBoost(SVM)` and so on to represent the various implementations of the classifiers. The experimental results of `AdaBoost` are not given here, since it is found in our experiments that `AdaBoost` can hardly improve the generalization error of `SVM` on all the data sets.

5.3. Experimental Results

Each same-distribution data set is split into two sets: a same-distribution training set T_s and a test set S. Table 3 presents the experimental results of `SVM`, `SVMt`, `AUX` and `TrAdaBoost(SVM)` when the ratio between same-distribution and diff-distribution training data is 0.01. Table 4 presents the experimental results of `TSVM`, `TSVMt` and `TrAdaBoost(TSVM)` when there are very few same-distribution training instances (one positive and one negative). The performance in error rate was the average of 10 repeats by random. The number of iterations is set to 100.

From Tables 3 and 4, we can see that the error rates given by `TrAdaBoost(SVM)` (or `TrAdaBoost(TSVM)`) are strictly lower than those given by `SVM` (or `TSVM`), `SVMt` (or `TSVMt`) and `AUX`. Intuitively, this is true because `SVM` is not a learning technique designed

Table 4. The error rates when semi-supervised learning

Data Set	`TSVM`	`TSVMt`	`TrAdaBoost(TSVM)`
rec vs talk	0.059	0.040	**0.021**
rec vs sci	0.067	0.062	**0.013**
sci vs talk	0.173	0.106	**0.075**
auto vs aviation	0.043	0.103	**0.038**
real vs simulated	0.144	0.131	**0.102**
orgs vs people	0.358	0.292	**0.248**
orgs vs places	0.424	0.436	**0.304**
people vs places	0.307	0.225	**0.179**
edible vs poisonous	0.439	0.179	**0.160**

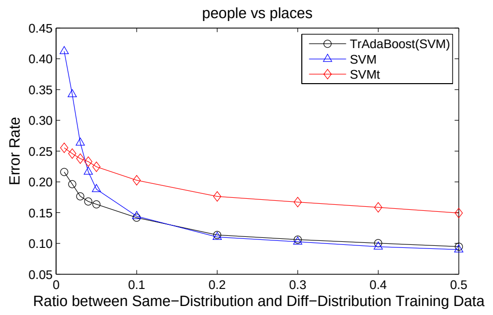

Figure 1. The error rate curves on `people vs places` data set for three classifiers `TrAdaBoost(SVM)`, `SVM` and `SVMt`

for transfer classification, but `TrAdaBoost` is. However, as several researchers have noted, transfer learning does not always reduce the generalization error and sometimes even lower the performance on the test set (e.g. Caruana (1997)). They call the phenomenon that transfer learning lower the original performance *negative transfer*. Although in our experiments, `TrAdaBoost` always gives better or comparative performances than the baselines, `TrAdaBoost` does not guarantee to improve the basic learner either.

In Figure 1, we focus on the `people vs places` data set. The ratio between same-distribution and diff-distribution training examples is gradually increased from 0.01 to 0.5. It can be seen that `TrAdaBoost(SVM)` always improves the performance of `SVMt`. `TrAdaBoost(SVM)` also outperforms `SVM`, when the ratio is not very large (i.e. less than 0.1). But, when the ratio is larger than 0.2, the performance of `TrAdaBoost(SVM)` is a little worse than `SVM`, but still comparative. We believe that diff-distribution training data contain not only good knowledge, but also noisy data. When there are too few same-distribution training data to train a good classifier, the useful knowledge from diff-distribution training data may help the learner, while the noisy part of the data does not affect the learner too much.

From Figure 1, we can see that a main contribution of `TrAdaBoost` is in situations when the ratio between same-distribution and diff-distribution training data is less than 0.1. When the ratio is greater than 0.2, the same-distribution training data might be sufficient for supervised learning. However, labeling such amount of same-distribution data are quite expensive.

Figure 2 presents the error rates on another data set `orgs vs places`. The curves are quite different from those in Figure 1. In Figure 2, the improvement of `TrAdaBoost` is much smaller, because the diff-distribution training data in `orgs vs places` data set are not as good for transfer learning the same-

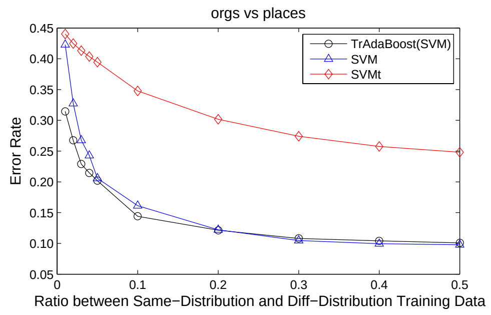

Figure 2. The error rate curves on `orgs vs places` data set for three classifiers `TrAdaBoost(SVM)`, `SVM` and `SVMt`

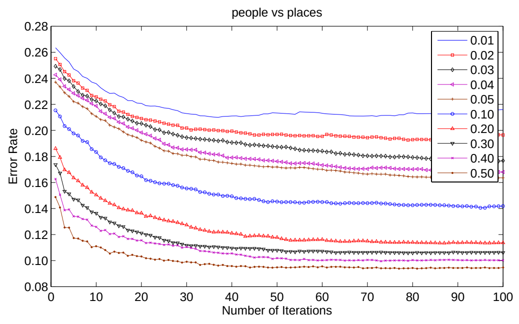

Figure 3. Iteration curves by error rate on the `people vs places` data set, where different curves indicate the cases under different ratios between same-distribution and diff-distribution training data

distribution knowledge. The evidence is that the error rate curve of `SVMt` in Figure 2 is a bit worse than that in Figure 1. `SVMt` is a straightforward learning method based on both same-distribution and diff-distribution training data. The performance of `SVMt` in Figure 2 indicates the quality of diff-distribution training data is low. Due to the low quality of the diff-distribution training data, `TrAdaBoost(SVM)` can only give comparable results with `SVM` on the `orgs vs places` data set.

Figure 3 shows the error rate results as a function of the number of iterations on the `people vs places` data set. Although the curves are not quite smooth, they converge well, which accords with the theoretical analysis in Section 4 and the past results about `AdaBoost` (Schapire et al., 1997). But, this is accompanied with a low rate of convergence. In Figure 3, `TrAdaBoost` does not converge well until at least 50 iterations.

Finally, we test how the difference in the distribution between training and test data influence the performance of `TrAdaBoost`. For each data set, we calculate the KL-divergence as well as the relative improvement by error rate reduction between `TrAdaBoost(SVM)` and `SVM` (or `SVMt`) in Figure 4. The data sets have been sorted by KL-divergence in increasing order from left to right. In this figure, the rate of improvement by `TrAdaBoost(SVM)` over `SVMt` increases along with the increment of KL-divergence, by and large. However, the improvement over `SVM` seems irregular.

Regarding the reason to the relative improvement by `TrAdaBoost(SVM)` over `SVM` being irregular, we observed that, when the number of iterations is set to 1, `TrAdaBoost(SVM)` becomes `SVMt`. In our opinion, the main improvement of `TrAdaBoost(SVM)` is based on `SVMt`. `SVM` only considers the information from same-distribution data, which is different from `TrAdaBoost(SVM)` and `SVMt`. Thus, we believe it is not easy to find the relation between the distribution distance and the relative improvement by `TrAdaBoost(SVM)` over `SVM`.

6. Conclusions and Future Work

In this paper, we proposed a novel framework `TrAdaBoost` for transferring knowledge from one distribution to another by boosting a basic learner. The basic idea is to select the most useful diff-distribution instances as additional training data for predicting the labels of same-distribution techniques. The theoretical analysis shows that `TrAdaBoost` first obeys the same-distribution training data, and then chooses the most helpful diff-distribution training instances as additional training data. Moreover, in our experiments, `TrAdaBoost` also demonstrates better transfer ability than traditional learning techniques. In almost all situations, `TrAdaBoost` gives better performance than the baseline methods.

We note that although we have proved the convergence of the prediction error on the same-distribution data, the improvement that we obtain is sensitive to the quality (or KL-divergence) of diff-distribution examples. This issue needs further understanding. Also, the rate of convergence ($O(\sqrt{\ln n/N})$) of `TrAdaBoost` may be slow. In addition, `TrAdaBoost` is able to transfer knowledge from only one distribution at one time, and cannot deal with multiple different distributions simultaneously. In the future, we will try to extend the framework to address these issues.

Acknowledgements

Wenyuan Dai, Gui-Rong Xue and Yong Yu are supported by a grant from National Natural Science Foundation of China (NO. 60473122). Qiang Yang is supported by a grant from Hong Kong research grant (HK-RGC 621606). We thank the anonymous reviewers for their greatly helpful comments.

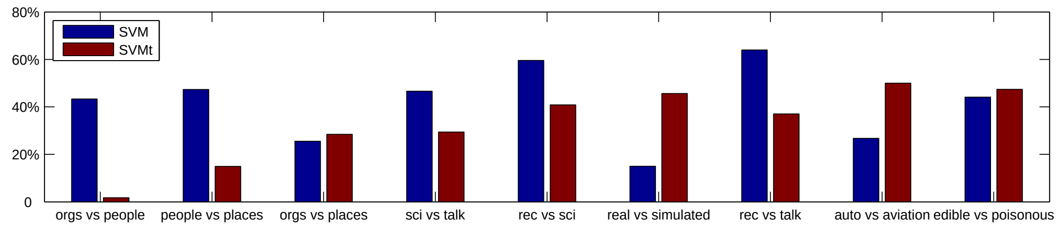

Figure 4. Error rate reductions on all data sets sorted by KL-divergence in increasing order from left to right

References

Ben-David, S., & Schuller, R. (2003). Exploiting task relatedness for multiple task learning. *Proceedings of the Sixteenth Annual Conference on Learning Theory.*

Bickel, S., & Scheffer, T. (2007). Dirichlet-enhanced spam filtering based on biased samples. In *Advances in neural information processing systems 19.*

Boser, B. E., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory.*

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28(1)*, 41–75.

DauméIII, H., & Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, *26*, 101–126.

Dudík, M., Schapire, R., & Phillips, S. (2006). Correcting sample selection bias in maximum entropy density estimation. In *Advances in neural information processing systems 18.*

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55(1)*, 119–139.

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica*, *47*, 153–161.

Huang, J., Smola, A., Gretton, A., Borgwardt, K. M., & Schölkopf, B. (2007). Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems 19.*

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Proceedings of Sixteenth International Conference on Machine Learning.*

Joachims, T. (2002). *Learning to classify text using support vector machines.* Dissertation, Kluwer.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, *22(1)*, 79–86.

Liao, X., Xue, Y., & Carin, L. (2005). Logistic regression with an auxiliary data source. *Proceedings of the Twenty-Second International Conference on Machine Learning.*

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (2005). To transfer or not to transfer. *Proceedings of NIPS 2005 Workshop on Inductive Transfer: 10 Years Later.*

Schapire, R. E. (1999). A brief introduction to boosting. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence.*

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. *Proceedings of the Fourteenth International Conference on Machine Learning.*

Schmidhuber, J. (1994). *On learning how to learn learning strategies* (Technical Report FKI-198-94). Fakultat fur Informatik.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, *90*, 227–244.

Thrun, S., & Mitchell, T. M. (1995). Learning one more thing. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence.*

Wu, P., & Dietterich, T. G. (2004). Improving svm accuracy by training on auxiliary data sources. *Proceedings of the Twenty-First International Conference on Machine Learning.*

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. *Proceedings of the Twenty-First International Conference on Machine Learning.*