Domain Adaptation via Transfer Component Analysis — Source link

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, Qiang Yang

Institutions: Nanyang Technological University, Hong Kong University of Science and Technology

Published on: 01 Feb 2011 - IEEE Transactions on Neural Networks (IEEE)

Topics: Domain (software engineering), Dimensionality reduction, Semi-supervised learning, Kernel method and Feature extraction

Related papers:

- A Survey on Transfer Learning
- Geodesic flow kernel for unsupervised domain adaptation
- Transfer Feature Learning with Joint Distribution Adaptation
- Adapting visual category models to new domains
- Unsupervised Visual Domain Adaptation Using Subspace Alignment
Domain Adaptation via Transfer Component Analysis

Sinno Jialin Pan¹, Ivor W. Tsang², James T. Kwok¹ and Qiang Yang¹
¹Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
²School of Computer Engineering, Nanyang Technological University, Singapore 639798
¹{sinnopan,jamesk,qyang}@cse.ust.hk, ²ivortsang@ntu.edu.sg

Abstract

Domain adaptation solves a learning problem in a target domain by utilizing the training data in a different but related source domain. Intuitively, discovering a good feature representation across domains is crucial. In this paper, we propose to find such a representation through a new learning method, transfer component analysis (TCA), for domain adaptation. TCA tries to learn some transfer components across domains in a Reproducing Kernel Hilbert Space (RKHS) using Maximum Mean Discrepancy (MMD). In the subspace spanned by these transfer components, data distributions in different domains are close to each other. As a result, with the new representations in this subspace, we can apply standard machine learning methods to train classifiers or regression models in the source domain for use in the target domain. The main contribution of our work is that we propose a novel feature representation in which to perform domain adaptation via a new parametric kernel using feature extraction methods, which can dramatically minimize the distance between domain distributions by projecting data onto the learned transfer components. Furthermore, our approach can handle large datasets and naturally lead to out-of-sample generalization. The effectiveness and efficiency of our approach in are verified by experiments on two real-world applications: cross-domain indoor WiFi localization and cross-domain text classification.

1 Introduction

Domain adaptation aims at adapting a classifier or regression model trained in a source domain for use in a target domain, where the source and target domains may be different but related. This is particularly crucial when labeled data are in short supply in the target domain. For example, in indoor WiFi localization, it is very expensive to calibrate a localization model in a large-scale environment. However, the WiFi signal strength may be a function of time, device or space, depending on dynamic factors. To reduce the re-calibration effort, we might want to adapt a localization model trained in one time period (the source domain) for a new time period (the target domain), or to adapt the localization model trained on one mobile device (the source domain) for a new mobile device (the target domain). However, the distributions of WiFi data collected over time or across devices may be very different, hence domain adaptation is needed [Yang et al., 2008]. Another example is sentiment classification. To reduce the effort of annotating reviews for various products, we might want to adapt a learning system trained on some types of products (the source domain) for a new type of product (the target domain). However, terms used in the reviews of different types of products may be very different. As a result, distributions of the data over different types of products may be different and thus domain adaptation is again needed [Blitzer et al., 2007].

A major computational problem in domain adaptation is how to reduce the difference between the distributions of source and target domain data. Intuitively, discovering a good feature representation across domains is crucial. A good feature representation should be able to reduce the difference in distributions between domains as much as possible, while at the same time preserving important (geometric or statistical) properties of the original data.

Recently, several approaches have been proposed to learn a common feature representation for domain adaptation [Daumé III, 2007; Blitzer et al., 2006]. Daumé III [2007] proposed a simple heuristic nonlinear mapping function to map the data from both source and target domains to a high-dimensional feature space, where standard machine learning methods are used to train classifiers. Blitzer et al. [2006] proposed the so-called structural correspondence learning (SCL) algorithm to induce correspondences among features from the different domains. This method depends on the heuristic selection of pivot features that appear frequently in both domains. Although it is experimentally shown that SCL can reduce the difference between domains based on the A-distance measure [Ben-David et al., 2007], the heuristic criterion of pivot feature selection may be sensitive to different applications. Pan et al. [2008] proposed a new dimensionality reduction method, Maximum Mean Discrepancy Embedding (MMDE), for domain adaptation. The motivation of MMDE is similar to our proposed work. It also aims at learning a shared latent space underlying the domains where distance between distributions can be reduced. However, MMDE suf-
fers from two major limitations: (1) MMDE is transductive, and does not generalize to out-of-sample patterns; (2) MMDE learns the latent space by solving a semi-definite program (SDP), which is a very expensive optimization problem.

In this paper, we propose a new feature extraction approach, called transfer component analysis (TCA), for domain adaptation. It tries to learn a set of common transfer components underlying both domains such that the difference in distributions of data in the different domains, when projected onto this subspace, can be dramatically reduced. Then, standard machine learning methods can be used in this subspace to train classifiers or regression models across domains. More specifically, if two domains are related to each other, there may exist several common components (or latent variables) underlying the original data, while others may not. Some of these components may capture the intrinsic structure under the original data, while others may not. Our goal is to discover those components that do not cause distribution change across the domains and capture the structure of the original data well. We will show in this paper that, compared to MMDE, TCA is much more efficient and can handle the out-of-sample extension problem.

The rest of the paper is organized as follows. Section 2 first describes the problem statement and preliminaries of domain adaptation. Our proposed method is presented in Section 3. We then review some related works in Section 4. In Section 5, we conduct a series of experiments on indoor WiFi localization and text classification. The last section gives some conclusive discussions.

In the sequel, $A > 0$ (resp. $A \succeq 0$) means that the matrix $A$ is symmetric and positive definite (pd) (resp. positive semidefinite (psd)). Moreover, the transpose of vector / matrix (in both the input and feature spaces) is denoted by the superscript $\top$, $A^\dagger$ is the pseudo-inverse of the matrix $A$, and $\text{tr}(A)$ denotes the trace of $A$.

2 Preliminaries of Domain Adaptation

In this paper, we focus on the setting where the target domain has no labeled training data, but has plenty of unlabeled data. We also assume that some labeled data $D_S$ are available in a source domain, while only unlabeled data $D_T$ are available in the target domain. We denote the source domain data as $D_S = \{(x_{S1}, y_{S1}), \ldots, (x_{Sn}, y_{Sn})\}$, where $x_{Si} \in \mathcal{X}$ is the input and $y_{Si} \in \mathcal{Y}$ is the corresponding output. Similarly, we denote the target domain data as $D_T = \{(x_{T1}, \ldots, x_{Tn})\}$, where the input $x_{Ti}$ is also in $\mathcal{X}$. Let $P(X_S)$ and $Q(X_T)$ (or $P$ and $Q$ for short) be the marginal distributions of $X_S$ and $X_T$, respectively. In general, $P$ and $Q$ can be different. Our task is then to predict the labels of data in the target domain. The key assumption in a typical domain adaptation setting is $P \neq Q$, but $P(Y_S|X_S) = P(Y_T|X_T)$.

2.1 Maximum Mean Discrepancy

Many criteria, such as the Kullback-Leibler (KL) divergence, can be used to estimate the distance between distributions. However, many of these criteria are parametric, since an intermediate density estimate is usually required. To avoid such a non-trivial task, a non-parametric distance estimate between distributions is more desirable. Recently, Borgwardt et al. [2006] proposed the Maximum Mean Discrepancy (MMD) as a relevant criterion for comparing distributions based on the Reproducing Kernel Hilbert Space (RKHS). Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_n\}$ be random variable sets with distributions $P$ and $Q$. The empirical estimate of the distance between $P$ and $Q$, as defined by MMD, is

$$\text{Dist}(X,Y) = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(x_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(y_i) \right\|_{\mathcal{H}}.$$ (1)

where $\mathcal{H}$ is a universal RKHS [Steinwart, 2001], and $\phi: \mathcal{X} \rightarrow \mathcal{H}$.

Therefore, the distance between distributions of two samples can be well-estimated by the distance between the means of the two samples mapped into a RKHS.

3 Transfer Component Analysis

Based on the inputs $\{x_{Si}\}$ and outputs $\{y_{Si}\}$ from the source domain, and the inputs $\{x_{Ti}\}$ from the target domain, our task is to predict the unknown outputs $\{y_{Ti}\}$ in the target domain. The general assumption in domain adaptation is that $\mathcal{X}$ is the desired nonlinear transformation be $\phi: \mathcal{X} \rightarrow \mathcal{H}$. Let $X'_S = \{x'_{Si}\} = \{\phi(x_{Si})\}$, $X'_T = \{x'_{Ti}\} = \{\phi(x_{Ti})\}$ and $X' = X'_S \cup X'_T$ be the transformed input sets from the source, target and combined domains, respectively. Then, we desire that $P(X'_S) = Q(X'_T)$.

Assuming that $\phi$ is the feature map induced by a universal kernel. As shown in Section 2.1, the distance between two distributions $P$ and $Q$ can be empirically measured by the squared distance between the empirical means of the two domains:

$$\text{Dist}(X'_S, X'_T) = \left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \phi(x_{Si}) - \frac{1}{n_2} \sum_{i=1}^{n_2} \phi(x_{Ti}) \right\|_{\mathcal{H}}^2.$$ (2)

Therefore, a desired nonlinear mapping $\phi$ can be found by minimizing this quantity. However, $\phi$ is usually highly nonlinear and a direct optimization of (2) can get stuck in poor local minima. We thus need to find a new approach, based on the following assumption.

The key assumption in the proposed domain adaptation setting is that $P \neq Q$, but $P(Y_S|\phi(X_S)) = P(Y_T|\phi(X_T))$ under a transformation mapping $\phi$ on the input.

In Section 3.1, we first revisit Maximum Mean Discrepancy Embedding (MMDE) which proposed to learn the kernel matrix $K$ corresponding to the nonlinear mapping $\phi$ by solving a SDP optimization problem. In Section 3.2, we then propose a factorization of the kernel matrix for MMDE. An efficient eigendecomposition algorithm for kernel learning and computational issues are discussed in Sections 3.3 and 3.4.
3.1 Kernel Learning for Domain Adaptation

Instead of finding the nonlinear transformation $\phi$ explicitly, Pan et al. [2008] proposed to transform this problem as a kernel learning problem. By virtue of the kernel trick, $(i.e., k(x_i, x_j) = \phi(x_i)\phi(x_j))$, the distance between the empirical means of the two domains in (2) can be written as:

$$\text{Dist}(X'_S, X'_T) = \text{tr}(KL),$$

where

$$K = \begin{bmatrix} K_{S,S} & K_{S,T} \\ K_{T,S} & K_{T,T} \end{bmatrix} \quad (4)$$

is a $(n_1 + n_2) \times (n_1 + n_2)$ kernel matrix, $K_{S,S}$, $K_{T,T}$ and $K_{S,T}$ respectively are the kernel matrices defined by $k$ on the data in the source domain, target domain, and cross domains; and $L = [L_{ij}] \geq 0$ with $L_{ij} = \frac{1}{n_1}$ if $x_i, x_j \in X_S$; $L_{ij} = \frac{1}{n_2}$ if $x_i, x_j \in X_T$; otherwise, $0$. In the transductive setting, learning the kernel $k(., .)$ can be solved by learning the kernel matrix $K$ instead. In [Pan et al., 2008], the resultant kernel learning problem is formulated as a semi-definite program (SDP). Principal Component Analysis (PCA) is then applied on the learned kernel matrix to find a low-dimensional latent space across domains. This is referred to as Maximum Mean Discrepancy Embedding (MMDE).

3.2 Parametric Kernel Map for Unseen Patterns

There are several limitations of MMDE. First, it is transductive and cannot generalize on unseen patterns. Second, the criterion (3) requires $K$ to be positive semi-definite and the resultant kernel learning problem has to be solved by expensive SDP solvers. Finally, in order to construct low-dimensional representations of $X'_S$ and $X'_T$, the obtained $K$ has to be further post-processed by PCA. This may potentially discard useful information in $K$.

In this paper, we propose an efficient method to find a non-linear mapping $\phi$ based on kernel feature extraction. It avoids the use of SDP and thus its high computational burden. Moreover, the learned kernel $k$ can be generalized to out-of-sample patterns directly. Besides, instead of using a two-step approach as in MMDE, we propose a unified kernel learning method which utilizes an explicit low-rank representation.

First, recall that the kernel matrix $K$ in (4) can be decomposed as $K = (KK^{-1/2})(K^{-1/2}K)$, which is often known as the empirical kernel map [Schölkopf et al., 1998]. Consider the use of a $(n_1 + n_2) \times m$ matrix $\hat{W}$ to transform the corresponding feature vectors to a $m$-dimensional space. In general, $m \ll n_1 + n_2$. The resultant kernel matrix\(^1\) is then

$$\hat{K} = (KK^{-1/2}\hat{W})(\hat{W}^TK^{-1/2}K) = K\hat{W}\hat{W}^TK, \quad (5)$$

where $\hat{W} = K^{-1/2}\hat{W} \in \mathbb{R}^{(n_1+n_2) \times m}$. In particular, the corresponding kernel evaluation of $k$ between any two patterns $x_i$ and $x_j$ is given by

$$\hat{k}(x_i, x_j) = k_{x_i}^T\hat{W}\hat{W}^Tk_{x_j}, \quad (6)$$

where $k_{x_i} = [k(x_1, x_i), \ldots, k(x_{n_1+n_2}, x_i)]^T \in \mathbb{R}^{n_1+n_2}$. Hence, the kernel $\hat{k}$ in (6) facilitates a readily parametric form for out-of-sample kernel evaluations.

Moreover, using the definition of $\hat{K}$ in (5), the distance between the empirical means of the two domains can be rewritten as:

$$\text{Dist}(X'_S, X'_T) = \text{tr}((K\hat{W}\hat{W}^TK)L) = \text{tr}(W^TKHKW). \quad (7)$$

3.3 Transfer Components Extraction

In minimizing criterion (7), a regularization term $\text{tr}(W^TK)$ is usually needed to control the complexity of $W$. As will be shown later in this section, this regularization term can avoid the rank deficiency of the denominator in the generalized eigendecomposition. The kernel learning problem for domain adaptation then reduces to:

$$\min_W \text{tr}(W^TKHKW) + \mu \text{tr}(W^TKLKWK),$$

s.t. $W^TKHKW = I, \quad (8)$

where $\mu$ is a trade-off parameter, $I \in \mathbb{R}^{m \times m}$ is the identity matrix, $H = I_{n_1+n_2} - \frac{1}{n_1+n_2}11^T$ is the centering matrix, where $1 \in \mathbb{R}^{n_1+n_2}$ is the column vector with all ones, and $I_{n_1+n_2} \in \mathbb{R}^{(n_1+n_2) \times (n_1+n_2)}$ is the identity matrix. Moreover, note that the constraint $W^TKHKW = I$ is added in (8) to avoid the trivial solution ($W = 0$), such that the transformed patterns do not collapse to one point, which can inflate the learned kernel $k$ such that the embedding of data $x'_i$ is preserved as in kernel PCA.

Though the optimization problem (8) involves a non-convex norm constraint $W^TKHKW = I$, it can still be solved efficiently by the following trace optimization problem:

**Proposition 1** The optimization problem (8) can be reformulated as

$$\min_W \text{tr}((W^TKHKW)^2W^T(I + \mu KWK)W), \quad (9)$$

or

$$\max_W \text{tr}((W^T(I + \mu KWK))W^{-1}W^TKHKW). \quad (10)$$

**Proof.** The Lagrangian of (8) is

$$\text{tr}(W^T(I + \mu KWK)W) - \text{tr}((W^TKHKW - I)Z), \quad (11)$$

where $Z$ is a symmetric matrix. Setting the derivative of (11) w.r.t. $W$ to zero, we have

$$(I + \mu KWK)W = KHKWZ. \quad (12)$$

Multiplying both sides on the left by $W^T$, and then on substituting it into (11), we obtain (9). Since the matrix $I + \mu KWK$ is non-singular benefited from the regularization term $\text{tr}(W^TK)$, we obtain an equivalent trace maximization problem (10).

**□**

Similar to kernel Fisher discriminant (KFD), the solution of $W$ in (10) is the eigenvectors corresponding to the $m$ leading eigenvalues of $(I + \mu KWK)^{-1}KHKW$, where at most $n_1+n_2-1$ eigenvectors can be extracted. In the sequel, the proposed method is referred to as Transfer Component Analysis (TCA).
3.4 Computational Issues

The kernel learning algorithm in [Pan et al., 2008] relies on SDPs. As there are $O((n_1 + n_2)^2)$ variables in $\tilde{K}$, the overall training complexity is $O((n_1 + n_2)^6.5)$ [Nesterov and Nemirovskii, 1994]. This becomes computationally prohibitive even for small-sized problems. Note that criterion (3) in this kernel learning problem is similar to the recently proposed supervised dimensionality reduction method colored MVU [Song et al., 2008], in which low-rank approximation is used to reduce the number of constraints and variables in the SDP. However, gradient descent is required to refine the embedding space and thus the solution can get stuck in a local minimum. On the other hand, our proposed kernel learning method requires only a simple and efficient eigendecomposition. This takes only $O(m(n_1 + n_2)^2)$ time when $m$ non-zero eigenvectors are to be extracted [Sorensen, 1996].

4 Related Works

Domain adaptation, which can be considered as a special setting of transfer learning [Pan and Yang, 2008], has been widely studied in natural language processing (NLP) [Ando and Zhang, 2005; Blitzer et al., 2006; Daumé III, 2007]. Ando and Zhang [2005] and Blitzer [2006] proposed structural correspondence learning (SCL) algorithms to learn the common feature representation across domains based on some heuristic selection of pivot features. Daumé III [2007] designed a heuristic kernel to augment features for solving some specific domain adaptation problems in NLP. Besides, domain adaptation has also been investigated in other application areas such as sentiment classification [Blitzer et al., 2007]. Theoretical analysis of domain adaptation has also been studied in [Ben-David et al., 2007].

The problem of sample selection bias (also referred to as co-variate shift) is also related to domain adaption. In sample selection bias, the basic assumption is that the sampling processes between the training data $X_{trn}$ and test data $X_{tst}$ may be different. As a result, $P(X_{trn}) \neq P(X_{tst})$, but $P(Y_{trn}|X_{trn}) = P(Y_{tst}|X_{tst})$. Instance re-weighting is a major technique for correcting sample selection bias [Huang et al., 2007; Sugiyama et al., 2008]. Recently, a state-of-the-art method, called kernel mean matching (KMM), is proposed [Huang et al., 2007]. It re-weights instances in a RKHS based on the MMD theory, which is different from our proposed method. Sugiyama et al. [2008] proposed another re-weighting algorithm, Kullback-Leibler Importance Estimation Procedure (KLIEP), which is integrated with cross-validation to perform model selection automatically. Xing et al. [2007] proposed to correct the labels predicted by a shift-unaware classifier towards a target distribution based on the mixture distribution of the training and test data. Matching distributions by re-weighting instances is also used successfully in Multi-task Learning [Bickel et al., 2008]. However, unlike instance re-weighting, the proposed TCA method can cope with noisy features (as in image data and WiFi data) by effectively denoising and finding a latent space for matching distributions across different domains simultaneously. Thus, TCA can be treated as an integration of unsupervised feature extraction and distribution matching in a latent space.

5 Experiments

In this section, we apply the proposed domain adaptation algorithm TCA on two real-world problems: indoor WiFi localization and text classification.

5.1 Cross-domain WiFi Localization

For cross-domain WiFi localization, we use a dataset published in the 2007 IEEE ICDM Contest [Yang et al., 2008]. This dataset contains some labeled WiFi data collected in time period A (the source domain) and a large amount of unlabeled WiFi data collected in time period B (the target domain). Here, a label means the corresponding location where the WiFi data are received. WiFi data collected from different time periods are considered as different domains. The task is to predict the labels of the WiFi data collected in time period B. More specifically, all the WiFi data are collected in an indoor building around $145.5 \times 37.5 \text{ m}^2$, 621 labeled data are collected in time period A and 3128 unlabeled data are collected in time period B.

We conduct a series of experiments to compare TCA with some baselines, including other feature extraction methods such as KPCA, sample selection bias (or co-variate shift) methods, KMM and KLIEP and a domain adaptation method, SCL. For each experiment, all labeled data in the source domain and some unlabeled data in the target domain are used for training. Evaluation is then performed on the remaining unlabeled data (out-of-sample) in the target domain. This is repeated 10 times and the average performance is used to measure the generalization abilities of the methods. In addition, to compare the performance between TCA and MMDE, we conduct some experiments in the transductive setting [Nigam et al., 2000]. The evaluation criterion is the Average Error Distance (AED) on the test data, and the lower the better. For determining parameters for each method, we randomly select a very small subset of the target domain data to tune parameters. The values of parameters are fixed for all the experiments.

Figure 1(a) compares the performance of Regularized Least Square Regression (RLSR) model on different feature representations learned by TCA, KPCA and SCL, and different re-weighted instances learned by KMM and KLIEP. Here, we use $\mu = 0.1$ for TCA and the Laplacian kernel. As can be seen, the performance can be improved with the new feature representations of TCA and KPCA. TCA can achieve much higher performance because it aims at finding the leading components that minimize the difference between domains. Then, from the space spanned by these components, the model trained in one domain can be used to perform accurate prediction in the other domain.

Figure 1(b) shows the results under a varying number of unlabeled data in the target main. As can be seen, with only a few unlabeled data in the target domain, TCA can still find a good feature representation to bridge between domains.

Since MMDE cannot generalize to out-of-sample patterns, in order to compare TCA with MMDE, we conduct another series of experiments in a transductive setting, which means that the trained models are only evaluated on the unlabeled data that are used for learning the latent space. In Figure 1(c), we apply MMDE and TCA on 621 labeled data from the
source domain and 300 unlabeled data from the target domain to learn new representations, respectively, and then train RLSR on them. More comparison results in terms of ACE with varying number of training data are shown in Table 1. The experimental results show that TCA is slightly higher (worse) than MMDE in terms of AED. This is due to the non-parametric kernel matrix learned by MMDE, which can fit the observed unlabeled data better. However, as mentioned in Section 3.4, the cost of MMDE is expensive due to the computationally intensive SDP. The comparison results between TCA and MMDE in terms of computational time on the WiFi dataset are shown in Table 2.

Table 1: ACE (in m) of MMDE and TCA with 10 dimensions and varying # training data (# labeled data in the source domain is fixed to 621, # unlabeled data in the target domain varies from 100 to 800.)

<table>
<thead>
<tr>
<th></th>
<th># unlabeled and labeled data used for training</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>721</td>
<td>821</td>
<td>921</td>
<td>1,021</td>
<td>1,121</td>
<td>1,221</td>
<td>1,321</td>
<td>1,421</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCA</td>
<td>2.413</td>
<td>2.378</td>
<td>2.313</td>
<td>2.285</td>
<td>2.274</td>
<td>2.285</td>
<td>2.287</td>
<td>2.289</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MMDE</td>
<td>2.315</td>
<td>2.247</td>
<td>2.208</td>
<td>2.212</td>
<td>2.207</td>
<td>2.182</td>
<td>2.237</td>
<td>2.279</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: CPU training time (in sec) of MMDE and TCA with varying # training data.

<table>
<thead>
<tr>
<th></th>
<th># unlabeled and labeled data used for training</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>721</td>
<td>821</td>
<td>921</td>
<td>1,021</td>
<td>1,121</td>
<td>1,221</td>
<td>1,321</td>
<td>1,421</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCA</td>
<td>25</td>
<td>30</td>
<td>46</td>
<td>59</td>
<td>72</td>
<td>94</td>
<td>115</td>
<td>145</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Cross-domain Text Classification

In this section, we perform cross-domain binary classification experiments on a preprocessed dataset of Reuters-21578. These data are categorized to a hierarchical structure. Data from different sub-categories under the same parent category are considered to be from different but related domains. The task is to predict the labels of the parent category. By following this strategy, three datasets **orgs vs people, orgs vs places** and **people vs places** are constructed. We randomly select 50% labeled data from the source domain, and 35% unlabeled data from the target domain. Evaluation is based on the (out-of-sample) testing of the remaining 65% unlabeled data in the target domain. This is repeated 10 times and the average results reported.

As can be seen from Table 3, different from experiments on the WiFi data, sample selection bias methods, such as KMM and KLIEP perform better than KPCA and PCA on the text data. However, with the feature presentations learned by TCA, SVM performs the best for cross-domain classification. This is because TCA not only discovers latent topics behind the text data, but also matches distributions across domains in the latent space spanned by the latent topics. Moreover, the performance of TCA using the RBF kernel is more stable.

6 Conclusion and Future Work

Learning feature representations is of primarily an important task for domain adaptation. In this paper, we propose a new feature extraction method, called Transfer Component Analysis (TCA), to learn a set of transfer components which reduce the distance across domains in a RKHS. Compared to the previously proposed MMDE for the same task, TCA is much more efficient and can be generalized to out-of-sample patterns. Experiments on two real-world datasets verify the effectiveness of the proposed method. In the future, we are planning to take side information into account when learning the transfer components across domains, which may be better for the final classification or regression tasks.

7 Acknowledgement

Sinno Jialin Pan and Qiang Yang thank the support from Microsoft Research MRA07/08.EG01 and Hong Kong CERG Project 621307. Ivor W. Tsang thanks the support from Singapore MOE AcRF Tier-1 Research Grant (RG15/08). James T. Kwok thanks the support from CERG project 614508.
Table 3: Comparison between Different Methods (number inside parentheses is the standard deviation over 10 repetitions).

<table>
<thead>
<tr>
<th>features</th>
<th>#features</th>
<th>people vs places</th>
<th>orgs vs people</th>
<th>orgs vs places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>5</td>
<td>0.5198 (.0252)</td>
<td>0.6696 (.0287)</td>
<td>0.6683 (.0221)</td>
</tr>
<tr>
<td>PCA</td>
<td>5</td>
<td>0.5198 (.0788)</td>
<td>0.3574 (.0760)</td>
<td>0.5653 (.0984)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.5453 (.0911)</td>
<td>0.6470 (.0598)</td>
<td>0.6140 (.0534)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.5424 (.0590)</td>
<td>0.6703 (.0334)</td>
<td>0.6491 (.0391)</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.5631 (.0346)</td>
<td>0.6652 (.0549)</td>
<td>0.6114 (.0564)</td>
</tr>
<tr>
<td>KPCA (RBF)</td>
<td>5</td>
<td>0.5900 (.0185)</td>
<td>0.5955 (.0676)</td>
<td>0.6267 (.0184)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.5934 (.0169)</td>
<td>0.5955 (.0676)</td>
<td>0.6267 (.0184)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.6032 (.0323)</td>
<td>0.5968 (.0705)</td>
<td>0.6908 (.0315)</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.6000 (.0267)</td>
<td>0.5964 (.0742)</td>
<td>0.6247 (.0438)</td>
</tr>
<tr>
<td>TCA (linear)</td>
<td>5</td>
<td>0.5804 (.0528)</td>
<td>0.6397 (.0897)</td>
<td>0.6403 (.0722)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.5495 (.0764)</td>
<td>0.7308 (.0495)</td>
<td>0.7006 (.0527)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.5600 (.0969)</td>
<td>0.7425 (.0579)</td>
<td>0.6720 (.0374)</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.5468 (.0635)</td>
<td>0.7330 (.0432)</td>
<td>0.5989 (.0700)</td>
</tr>
<tr>
<td>TCA (RBF)</td>
<td>5</td>
<td>0.6129 (.0176)</td>
<td>0.6297 (.0302)</td>
<td>0.6899 (.0195)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.5920 (.0148)</td>
<td>0.7088 (.0251)</td>
<td>0.7042 (.0218)</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.5954 (.0201)</td>
<td>0.7196 (.0235)</td>
<td>0.6942 (.0220)</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.5916 (.0166)</td>
<td>0.7217 (.0275)</td>
<td>0.6896 (.0203)</td>
</tr>
<tr>
<td>SCL</td>
<td></td>
<td>0.3267 (.0310)</td>
<td>0.6834 (.0327)</td>
<td>0.6733 (.0198)</td>
</tr>
<tr>
<td>KMM (linear)</td>
<td></td>
<td>0.3836 (.0139)</td>
<td>0.7006 (.0353)</td>
<td>0.6114 (.0263)</td>
</tr>
<tr>
<td>KMM (RBF)</td>
<td></td>
<td>0.3836 (.0139)</td>
<td>0.6968 (.0224)</td>
<td>0.6655 (.0245)</td>
</tr>
<tr>
<td>KLIEP</td>
<td></td>
<td>0.3758 (.0241)</td>
<td>0.6946 (.0192)</td>
<td>0.6638 (.0112)</td>
</tr>
</tbody>
</table>

References


