Supervised Representation Learning: Transfer Learning with Deep Autoencoders

Fuzhen Zhuang[1], **Xiaohu Cheng**[1,2], **Ping Luo**[1], **Sinno Jialin Pan**[3], **Qing He**[1]
[1]Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. {zhuangfz, heq}@ics.ict.ac.cn, luop@ict.ac.cn
[2]University of Chinese Academy of Sciences, Beijing, China. chengxh@ics.ict.ac.cn
[3]Nanyang Technological University, Singapore 639798. sinnopan@ntu.edu.sg

Abstract

Transfer learning has attracted a lot of attention in the past decade. One crucial research issue in transfer learning is how to find a good representation for instances of different domains such that the divergence between domains can be reduced with the new representation. Recently, deep learning has been proposed to learn more robust or higher-level features for transfer learning. However, to the best of our knowledge, most of the previous approaches neither minimize the difference between domains explicitly nor encode label information in learning the representation. In this paper, we propose a supervised representation learning method based on deep autoencoders for transfer learning. The proposed deep autoencoder consists of two encoding layers: an embedding layer and a label encoding layer. In the embedding layer, the distance in distributions of the embedded instances between the source and target domains is minimized in terms of KL-Divergence. In the label encoding layer, label information of the source domain is encoded using a softmax regression model. Extensive experiments conducted on three real-world image datasets demonstrate the effectiveness of our proposed method compared with several state-of-the-art baseline methods.

1 Introduction

Transfer learning focuses on adapting knowledge from an auxiliary source domain to a target domain with little or without any label information to build a target prediction model of good generalization performance. In the past decade, a lot of attention has been paid on developing methods to transfer knowledge effectively across domains [Pan and Yang, 2010]. A crucial research issue in transfer learning is how to reduce difference between the source and target domains while preserving original data properties. Among different approaches to transfer learning, the feature-based transfer learning methods have proven to be superior for the scenarios where original raw data between domains are very different while the divergence between domains can be reduced. A common objective of feature-based transfer learning methods is to learn a transformation to project instances from different domains to a common latent space where the difference of the projected instances between domains can be reduced [Blitzer *et al.*, 2006; Dai *et al.*, 2007a; Pan *et al.*, 2008; 2011; Zhuang *et al.*, 2014].

Recently, because of the power on learning high-level features, deep learning has been applied to transfer learning [Xavier and Bengio, 2011; Chen *et al.*, 2012; Joey Tianyi Zhou and Yan, 2014]. Xavier and Bengio [2011] proposed to learn robust features with stacked denoising autoencoders (SDA) [Vincent *et al.*, 2010] on the union of data of a number of domains. The learned new features are considered as high-level features, and used to represent both the source and target domain data. Finally, standard classifiers, e.g., support vector machines (SVMs), are trained on the source domain labeled data with the new representations, and make predictions on the target domain data with the new representations. Chen *et al.* [2012] extended the work of SDA, and proposed the marginalized SDA (mSDA) for transfer learning. mSDA addresses two limitations of SDA: highly computational cost and lack of scalability with high-dimensional features. More recently, Joey Tianyi Zhou and Yan [2014] proposed a deep learning approach to heterogeneous transfer learning based on an extension of mSDA, where instances in the source and target domains are represented by heterogeneous features. In their proposed method, the bridge between the source and target domains with heterogeneous features is built based on the corresponding information of instances between the source and target domains, which is assumed to be given in advance.

Though the goal of previous deep-learning-based methods for transfer learning is to learn a more powerful feature representation to reduce the difference between domains, most of them did not explicitly minimize the distance between domains when learning the representation. Therefore, the reduction in difference between domains is not guaranteed with the learned feature representation. Furthermore, most previous methods are unsupervised, and thus fail to encode discriminative information into the representation learning.

In this paper, we propose a supervised representation learning method for transfer learning based on deep autoencoders. Specifically, the proposed method, named Transfer Learning with Deep Autoconders (TLDA), is shown in Figure 1. In

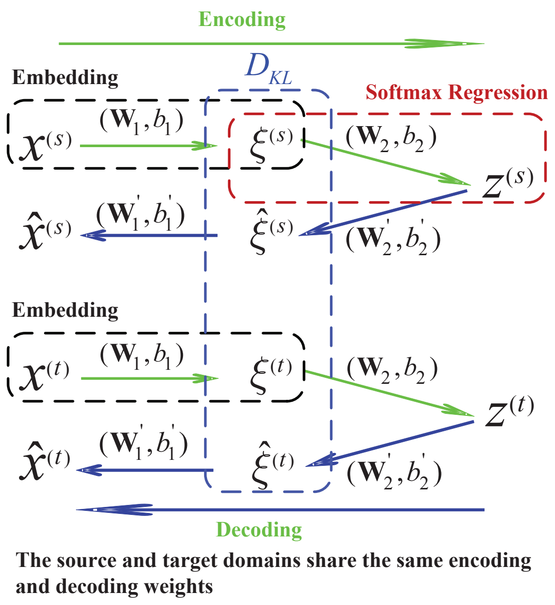


Figure 1: The framework of TLDA

TLDA, there are two encoding and decoding layers, respectively, where the encoding and decoding weights are shared by both the source and target domains. The first encoding layer is referred to as the embedding layer, where the distributions of the source- and target- domain data are enforced to be similar by minimizing the KL divergence [Kullback, 1987] of the embedded instances between domains. The second encoding layer is referred to as the label encoding layer, where the source domain label information is encoded using a softmax regression model [Friedman and Rob, 2010], which can naturally handle multiple classes. Note that, in the second encoding layer, the encoding weights are also used for the final classification model. In summary, there are three key features in our proposed TLDA:

1. The encoding and decoding weights are shared across different domains for knowledge transfer.
2. The distributions of two domains are enforced to be similar in the embedding space.
3. The label information is encoded.

2 Preliminary Knowledge

In this section, we first review some preliminary knowledge that is used in our proposed framework. Note that frequently used notations are listed in Table 1, and unless otherwise specified, all the vectors are column vectors.

2.1 Autoencoders

The basic framework of autoencoder [Bengio, 2009] is a feed forward neural network with an input layer, an output layer and one or more hidden layers between them. An autoencoder framework usually includes the encoding and decoding processes. Given an input $\boldsymbol{x}$, autoencoder first encodes it

Table 1: The Notation and Denotation

$\mathcal{D}_s, \mathcal{D}_t$	The source and target domains
n_s	The number of instances in source domain
n_t	The number of instances in target domain
m	The number of original features
k	The number of nodes in embedding layer
c	The number of nodes in label layer
$\boldsymbol{x}_i^{(s)}$, $\boldsymbol{x}_i^{(t)}$	The i-th instance of source and target domains
$\hat{\boldsymbol{x}}_i^{(s)}$, $\hat{\boldsymbol{x}}_i^{(t)}$	The reconstructions of $\boldsymbol{x}_i^{(s)}$ and $\boldsymbol{x}_i^{(t)}$
$y_i^{(s)}$	The label of instance $\boldsymbol{x}_i^{(s)}$
$\boldsymbol{\xi}_i^{(s)}$, $\boldsymbol{\xi}_i^{(t)}$	The hidden representations of $\boldsymbol{x}_i^{(s)}$ and $\boldsymbol{x}_i^{(t)}$
$\hat{\boldsymbol{\xi}}_i^{(s)}$, $\hat{\boldsymbol{\xi}}_i^{(t)}$	The reconstructions of $\boldsymbol{\xi}_i^{(s)}$ and $\boldsymbol{\xi}_i^{(t)}$
$\boldsymbol{z}_i^{(s)}$, $\boldsymbol{z}_i^{(t)}$	The hidden representations of $\boldsymbol{\xi}_i^{(s)}$ and $\boldsymbol{\xi}_i^{(t)}$
$\boldsymbol{W}_i$, $\boldsymbol{b}_i$	Encoding weight and bias matrix for layer i
$\boldsymbol{W}_i'$, $\boldsymbol{b}_i'$	Decoding weight and bias matrix for layer i
$\top$	The transposition of a matrix
$\circ$	The dot product of vectors or matrixes

to one or more hidden layers through several encoding processes, then decodes the hidden layers to obtain an output $\hat{\boldsymbol{x}}$. Autoencoder tries to minimize the deviation of $\hat{\boldsymbol{x}}$ from the input $\boldsymbol{x}$, and the process of autoencoder with one hidden layer can be summarized as:

$$\text{Encoding}: \boldsymbol{\xi} = f(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) \tag{1}$$

$$\text{Decoding}: \hat{\boldsymbol{x}} = f(\boldsymbol{W}_1'\boldsymbol{\xi} + \boldsymbol{b}_1') \tag{2}$$

where f is a nonlinear activation function (the sigmoid function is adopted in this paper), $\boldsymbol{W}_1 \in \mathbb{R}^{k\times m}$ and $\boldsymbol{W}_1' \in \mathbb{R}^{m\times k}$ are weight matrices, $\boldsymbol{b}_1 \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{b}_1' \in \mathbb{R}^{m\times 1}$ are bias vectors, and $\boldsymbol{\xi} \in \mathbb{R}^{k\times 1}$ is the output of the hidden layer. Given a set of inputs $\{\boldsymbol{x}_i\}_{i=1}^n$, the reconstruction error can be computed by $\sum_{i=1}^n \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2$. The goal of autoencoder is to learn the weight matrices $\boldsymbol{W}_1$ and $\boldsymbol{W}_1'$, and the bias vectors $\boldsymbol{b}_1$ and $\boldsymbol{b}_1'$ by minimizing the reconstruction error as follows,

$$\min_{\boldsymbol{W}_1,\boldsymbol{b}_1,\boldsymbol{W}_1',\boldsymbol{b}_1'} \sum_{i=1}^n \|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\|^2 \tag{3}$$

2.2 Softmax Regression

The softmax regression model [Friedman and Rob, 2010] is a generalization of the logistic regression model for multi-class classification problems, where the class label y can take more than two values, i.e., $y \in \{1, 2, ..., c\}$ (where $c \geq 2$ is the number of class labels.). For a test instance $\boldsymbol{x}$, we can estimate the probabilities of each class that $\boldsymbol{x}$ belongs to as follows,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \begin{bmatrix} p(y_i = 1|\boldsymbol{x};\boldsymbol{\theta}) \\ p(y_i = 2|\boldsymbol{x};\boldsymbol{\theta}) \\ \vdots \\ p(y_i = c|\boldsymbol{x};\boldsymbol{\theta}) \end{bmatrix} = \frac{1}{\sum_{j=1}^c e^{\boldsymbol{\theta}_j^\top \boldsymbol{x}}} \begin{bmatrix} e^{\boldsymbol{\theta}_1^\top \boldsymbol{x}} \\ e^{\boldsymbol{\theta}_2^\top \boldsymbol{x}} \\ \vdots \\ e^{\boldsymbol{\theta}_c^\top \boldsymbol{x}} \end{bmatrix} \tag{4}$$

where $\sum_{j=1}^c e^{\boldsymbol{\theta}_j^\top \boldsymbol{x}}$ is a normalized term, and $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_c$ are the model parameters.

Given the training set $\{\boldsymbol{x_i}, y_i\}_{i=1}^n$, $y_i \in \{1, 2, ..., c\}$, the solution of softmax regression can be derived by minimizing the following optimization problem,

$$\min_{\boldsymbol{\theta}} \left(-\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} 1\{y_i = j\} \log \frac{e^{\boldsymbol{\theta}_j^\top \boldsymbol{x}_i}}{\sum_{l=1}^{c} e^{\boldsymbol{\theta}_l^\top \boldsymbol{x}_i}} \right), \qquad (5)$$

where $1\{\cdot\}$ is an indicator function, whose value is 1 if the expression is true, otherwise 0. Once the model is trained, one can compute the probability of instance $\boldsymbol{x}$ belonging to a label j using Eq. (4), and assign its class label as

$$y = \max_{j} \frac{e^{\boldsymbol{\theta}_j^\top \boldsymbol{x}}}{\sum_{l=1}^{c} e^{\boldsymbol{\theta}_l^\top \boldsymbol{x}}}. \qquad (6)$$

2.3 Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence [Kullback, 1987], also known as the relative entropy, is a non-symmetric measure of the divergence between two probability distributions. Given two probability distributions $\boldsymbol{P} \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{Q} \in \mathbb{R}^{k\times 1}$, the KL divergence of $\boldsymbol{Q}$ from $\boldsymbol{P}$ is the information lost when $\boldsymbol{Q}$ is used to approximate $\boldsymbol{P}$ [Liddle *et al.*, 2010], defined as $D_{KL}(\boldsymbol{P}||\boldsymbol{Q}) = \sum_{i=1}^{k} \boldsymbol{P}(i) \ln(\frac{\boldsymbol{P}(i)}{\boldsymbol{Q}(i)})$. In this paper, we adopt the symmetrized version of KL-divergence, $KL(\boldsymbol{P}, \boldsymbol{Q}) = D_{KL}(\boldsymbol{P}||\boldsymbol{Q}) + D_{KL}(\boldsymbol{Q}||\boldsymbol{P})$, to measure the divergence for classification problems, Smaller value of KL divergence indicates more similar of two distributions. Thus, we use the KL divergence to measure the difference between two data domains when they are embedded to the same latent space.

3 Transfer Learning with Deep Autoencoders

3.1 Problem Formalization

Given two domains $\boldsymbol{D}_s$, and $\boldsymbol{D}_t$, where $\boldsymbol{D}_s$=$\{\boldsymbol{x}_i^{(s)}, y_i^{(s)}\}|_{i=1}^{n_s}$ is the source domain labeled data with $\boldsymbol{x}_i^{(s)} \in \mathbb{R}^{m\times 1}$, and $y_i^{(s)} \in \{1, ..., c\}$, while $\boldsymbol{D}_t = \{\boldsymbol{x}_i^{(t)}\}|_{i=1}^{n_t}$ is the target domain with unlabeled data. Here, n_s and n_t are the numbers of instances in $\boldsymbol{D}_s$ and $\boldsymbol{D}_t$, respectively.

As shown in Figure 1, there are three factors to be taken into consideration for representation learning. Therefore, the objective to be minimized in our proposed learning framework for transfer learning can be formalized as follows,

$$\begin{aligned} \mathcal{J} = \quad & J_r(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \alpha \Gamma(\boldsymbol{\xi}^{(s)}, \boldsymbol{\xi}^{(t)}) + \beta \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}^{(s)}) \\ + \quad & \gamma \Omega(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{W}', \boldsymbol{b}'). \end{aligned} \qquad (7)$$

The first term of the objective is the reconstruction error for both source and target domain data, which can be defined as,

$$J_r(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{r\in\{s,t\}} \sum_{i=1}^{n_r} ||\boldsymbol{x}_i^{(r)} - \hat{\boldsymbol{x}}_i^{(r)}||^2, \qquad (8)$$

where

$$\boldsymbol{\xi}_i^{(r)} = f(\boldsymbol{W}_1 \boldsymbol{x}_i^{(r)} + \boldsymbol{b}_1), \quad \boldsymbol{z}_i^{(r)} = f(\boldsymbol{W}_2 \boldsymbol{\xi}_i^{(r)} + \boldsymbol{b}_2), \quad (9)$$

$$\hat{\boldsymbol{\xi}}_i^{(r)} = f(\boldsymbol{W}_2' \boldsymbol{z}_i^{(r)} + \boldsymbol{b}_2'), \quad \hat{\boldsymbol{x}}_i^{(r)} = f(\boldsymbol{W}_1' \hat{\boldsymbol{\xi}}_i^{(r)} + \boldsymbol{b}_1'). \quad (10)$$

The first hidden layer is called the embedding layer with an output $\boldsymbol{\xi} \in \mathbb{R}^{k\times 1}$ of k nodes ($k \leq m$), a weight matrix $\boldsymbol{W}_1 \in \mathbb{R}^{k\times m}$, and a bias vector $\boldsymbol{b}_1 \in \mathbb{R}^{k\times 1}$. The output of first layer is the input for the second hidden layer. The second hidden layer is called the label layer with an output $\boldsymbol{z} \in \mathbb{R}^{c\times 1}$ of c nodes (equals to the number of class label), a weight matrix $\boldsymbol{W}_2 \in \mathbb{R}^{c\times k}$, and a bias vector $\boldsymbol{b}_2 \in \mathbb{R}^{c\times 1}$. Here, the softmax Regression is used as the regularization item on source domain to incorporate label information. In addition, the output of the second layer is used as the prediction results for target domain. The third hidden layer $\hat{\boldsymbol{\xi}}$ is the reconstruction of the embedding layer with the corresponding weight matrix and bias vector $\hat{\boldsymbol{\xi}} \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{W}_2' \in \mathbb{R}^{k\times c}, \boldsymbol{b}_2' \in \mathbb{R}^{k\times 1}$. Finally, $\hat{\boldsymbol{x}}$ is the reconstruction of $\boldsymbol{x}$ with $\hat{\boldsymbol{x}} \in \mathbb{R}^{m\times 1}$, $\boldsymbol{W}_1' \in \mathbb{R}^{m\times k}$, and $\boldsymbol{b}_1' \in \mathbb{R}^{m\times 1}$.

The second term in the objective Eq. (7) is the KL divergence of embedded instances between the source and target domains, which can be written as

$$\Gamma(\boldsymbol{\xi}^{(s)}, \boldsymbol{\xi}^{(t)}) = D_{KL}(\boldsymbol{P}_s||\boldsymbol{P}_t) + D_{KL}(\boldsymbol{P}_t||\boldsymbol{P}_s), \qquad (11)$$

where

$$\boldsymbol{P}_s' = \frac{1}{n_s} \sum_{i=1}^{n_s} \boldsymbol{\xi}_i^{(s)}, \quad \boldsymbol{P}_s = \frac{\boldsymbol{P}_s'}{\sum \boldsymbol{P}_s'}, \qquad (12)$$

$$\boldsymbol{P}_t' = \frac{1}{n_t} \sum_{i=1}^{n_t} \boldsymbol{\xi}_i^{(t)}, \quad \boldsymbol{P}_t = \frac{\boldsymbol{P}_t'}{\sum \boldsymbol{P}_t'}. \qquad (13)$$

The goal of minimizing the KL divergence of the embedded instances between the source and target domains is to ensure the source and target data distributions to be similar in the embedding space.

The third term in the objective Eq. (7) is the loss function of softmax regression to incorporate the label information of the source domain into the embedding space. Specifically, this term can be formalized as follows,

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\xi}^{(s)}) = -\frac{1}{n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{c} 1\{y_i^{(s)} = j\} \log \frac{e^{\boldsymbol{\theta}_j^\top \boldsymbol{\xi}_i^{(s)}}}{\sum_{l=1}^{c} e^{\boldsymbol{\theta}_l^\top \boldsymbol{\xi}_i^{(s)}}},$$

where $\boldsymbol{\theta}_j^\top$ ($j \in \{1, ..., c\}$) is the j-th row of $\boldsymbol{W}_2$.

Finally, the last term in the objective Eq. (7) is an regularization on model parameters, which is defined as follows,

$$\begin{aligned} \Omega(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{W}', \boldsymbol{b}') = \quad & ||\boldsymbol{W}_1||^2 + ||\boldsymbol{b}_1||^2 + ||\boldsymbol{W}_2||^2 + ||\boldsymbol{b}_2||^2 \\ & + ||\boldsymbol{W}_1'||^2 + ||\boldsymbol{b}_1'||^2 + ||\boldsymbol{W}_2'||^2 + ||\boldsymbol{b}_2'||^2. \end{aligned}$$

The trade-off parameters α, β, and γ are positive constants to balance the effect of different terms to the overall objective.

3.2 Model Learning

The minimization problem of Eq. (7) with respect to $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_2$, $\boldsymbol{W}_2'$, $\boldsymbol{b}_2'$, $\boldsymbol{W}_1'$, and $\boldsymbol{b}_1'$ is an unconstrained optimization problem. To solve this problem, we adopt the gradient descent methods. For succinctness, we first introduce some intermediate variables as follows.

$$\begin{aligned} A_i^{(r)} &= \left(\hat{\boldsymbol{x}}_i^{(r)} - \boldsymbol{x}_i^{(r)}\right) \circ \hat{\boldsymbol{x}}_i^{(r)} \circ \left(1 - \hat{\boldsymbol{x}}_i^{(r)}\right), \\ B_i^{(r)} &= \hat{\boldsymbol{\xi}}_i^{(r)} \circ \left(1 - \hat{\boldsymbol{\xi}}_i^{(r)}\right), \\ C_i^{(r)} &= \boldsymbol{z}_i^{(r)} \circ \left(1 - \boldsymbol{z}_i^{(r)}\right), \\ D_i^{(r)} &= \boldsymbol{\xi}_i^{(r)} \circ \left(1 - \boldsymbol{\xi}_i^{(r)}\right). \end{aligned}$$

The partial derivatives of the objective Eq. (7) w.r.t. $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_2$, $\boldsymbol{W}_2^{'}$, $\boldsymbol{b}_2^{'}$, $\boldsymbol{W}_1^{'}$, and $\boldsymbol{b}_1^{'}$ can be computed as follows respectively,

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_1} = & \sum_{i=1}^{n_s} 2\boldsymbol{W}_1^{'\top} A_i^{(s)} \circ (\boldsymbol{W}_2^{\top}(\boldsymbol{W}_2^{'\top} B_i^{(s)} \circ C_i^{(s)})) \circ D_i^{(s)} \boldsymbol{x}_i^{(s)\top} \\
& + \sum_{i=1}^{n_t} 2\boldsymbol{W}_1^{'\top} A_i^{(t)} \circ (\boldsymbol{W}_2^{\top}(\boldsymbol{W}_2^{'\top} B_i^{(t)} \circ C_i^{(t)})) \circ D_i^{(t)} \boldsymbol{x}_i^{(t)\top} \\
& + \frac{\alpha}{n_s} \sum_{i=1}^{n_s} D_i^{(s)} \circ (1 - \frac{\boldsymbol{P}_t}{\boldsymbol{P}_s} + ln(\frac{\boldsymbol{P}_s}{\boldsymbol{P}_t})) \boldsymbol{x}_i^{(s)\top} \qquad (14) \\
& + \frac{\alpha}{n_t} \sum_{i=1}^{n_t} D_i^{(t)} \circ (1 - \frac{\boldsymbol{P}_s}{\boldsymbol{P}_t} + ln(\frac{\boldsymbol{P}_t}{\boldsymbol{P}_s})) \boldsymbol{x}_i^{(t)\top} + 2\gamma \boldsymbol{W}_1 \\
& - \frac{\beta}{n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{c} 1\{y_i^{(s)} = j\} (\boldsymbol{W}_{2j}^{\top} - \frac{\boldsymbol{W}_2^{\top} e^{\boldsymbol{W}_2 \boldsymbol{\xi}_i^{(s)}}}{\sum_l e^{\boldsymbol{W}_{2l} \boldsymbol{\xi}_i^{(s)}}}) \\
& \circ D_i^{(s)} \boldsymbol{x}_i^{(s)\top}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_{2j}} = & \sum_{i=1}^{n_s} 2\boldsymbol{W}_{2j}^{'\top} (\boldsymbol{W}_1^{'\top} A_i^{(s)} \circ B_i^{(s)}) \circ C_{ij}^{(s)} \boldsymbol{\xi}_i^{(s)\top} \\
& + \sum_{i=1}^{n_t} 2\boldsymbol{W}_{2j}^{'\top} (\boldsymbol{W}_1^{'\top} A_i^{(t)} \circ B_i^{(t)}) \circ C_{ij}^{(t)} \boldsymbol{\xi}_i^{(t)\top} \qquad (15) \\
& - \frac{\beta}{n_{sj}} (\sum_{i=1}^{n_{sj}} \boldsymbol{\xi}_i^{(s)\top} - \sum_{i=1}^{n_s} \frac{e^{\boldsymbol{W}_{2j} \boldsymbol{\xi}_i^{(s)}}}{\sum_l e^{\boldsymbol{W}_{2l} \boldsymbol{\xi}_i^{(s)}}} \boldsymbol{\xi}_i^{(s)\top}) + 2\gamma \boldsymbol{W}_{2j},
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_2^{'}} = & \sum_{i=1}^{n_s} 2\boldsymbol{W}_1^{'\top} A_i^{(s)} \circ B_i^{(s)} \boldsymbol{z}_i^{(s)\top} + 2\gamma \boldsymbol{W}_2^{'} \\
& + \sum_{i=1}^{n_t} 2\boldsymbol{W}_1^{'T} A_i^{(t)} \circ B_i^{(t)} \boldsymbol{z}_i^{(t)\top}, \qquad (16)
\end{aligned}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_1^{'}} = \sum_{i=1}^{n_s} 2A_i^{(s)} \hat{\boldsymbol{\xi}}_i^{(s)\top} + \sum_{i=1}^{n_t} 2A_i^{(t)} \hat{\boldsymbol{\xi}}_i^{(t)\top} + 2\gamma \boldsymbol{W}_1^{'}, \qquad (17)$$

where $\boldsymbol{W}_{2j}$ is the j-th row of $\boldsymbol{W}_2$, and n_{sj} is the number of instance with the label j in source domain. As the partial derivatives of the objective Eq.(7) w.r.t. $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_2^{'}$, $\boldsymbol{b}_1^{'}$ are very similar to those of $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{W}_2^{'}$, $\boldsymbol{W}_1^{'}$, respectively, we omit the details due to the limit of space. Based on the above partial derivatives, we develop an alternatively iterating algorithm to derive the solutions by using the following rules,

$$\begin{aligned}
\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_1}, \quad & \boldsymbol{b}_1 \leftarrow \boldsymbol{b}_1 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_1}, \\
\boldsymbol{W}_1^{'} \leftarrow \boldsymbol{W}_1^{'} - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_1^{'}}, \quad & \boldsymbol{b}_1^{'} \leftarrow \boldsymbol{b}_1^{'} - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_1^{'}}, \\
\boldsymbol{W}_2 \leftarrow \boldsymbol{W}_2 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_2}, \quad & \boldsymbol{b}_2 \leftarrow \boldsymbol{b}_2 - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_2}, \qquad (18) \\
\boldsymbol{W}_2^{'} \leftarrow \boldsymbol{W}_2^{'} - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{W}_2^{'}}, \quad & \boldsymbol{b}_2^{'} \leftarrow \boldsymbol{b}_2^{'} - \eta \frac{\partial \mathcal{J}}{\partial \boldsymbol{b}_2^{'}},
\end{aligned}$$

where η is the step length, which determines the speed of convergence. The details of the proposed algorithm is summarized in Algorithm 1. Note that the proposed optimization problem is not convex, and thus there is no guarantee on obtaining an optimal global solution. To achieve a better local optimal solution of the proposed gradient descent approach, we first run SAE on all source and target domain data, and then use the output of SAE to initialize the encoding and decoding weights.

Algorithm 1 Transfer Learning with Deep Autoencoders (TLDA)

```
Input: Given one source domain $D_s = \{x_i^{(s)}, y_i^{(s)}\}|_{i=1}^{n_s}$, and one target domain $D_t = \{x_i^{(t)}\}|_{i=1}^{n_t}$, trade-off parameters $\alpha$, $\beta$, $\gamma$, the number of nodes in embedding layer and label layer, $k$ and $c$.
Output: Results of label layer $z$ and embedded layer $\xi$.
1. Initialize $W_1$, $W_2$, $W_2^{'}$, $W_1^{'}$ and $b_1$, $b_2$, $b_2^{'}$, $b_1^{'}$ by Stacked Autoencoders performed on both source and target domains;
2. Compute the partial derivatives of all variables according to Eqs. (14), (15) (16) and (17);
3. Iteratively update the variables using Eq. (18);
4. Continue Step2 and Step3 until the algorithm converges;
5. Computing the embedding layer $\xi$ and label layer $z$ using (9), and then construct target classifiers as described in Section 3.3.
```

3.3 Classifier Construction

After all the parameters are learned, we can construct classifiers for the target domain in two ways. The first way is directly to use the output of the second hidden layer. That is, for any instance $\boldsymbol{x}^{(t)}$ in the target domain, the output of the label layer $\boldsymbol{z}^{(t)} = f(\boldsymbol{W}_2 \boldsymbol{\xi}^{(t)} + \boldsymbol{b}_2)$ can indicate the probabilities of $\boldsymbol{x}^{(t)}$ which class it belongs to. We choose the maximum probability and the corresponding label as the prediction. The second way is to apply standard classification algorithms, e.g., logistic regression(LR) [Snyman, 2005; Friedman and Rob, 2010] to train a classifier for source domain in the embedding space. Then the classifier is applied to predict class labels for target domain data. These two methods are denoted as TLDA_1 and TLDA_2, respectively.

4 Experimental Evaluation

In this section, we conduct extensive experiments on three real-world image data sets to show the effectiveness of the proposed framework. Two of the three datasets are on binary classification, and the rest one is on multi-class classification.

4.1 Datasets and Preprocessing

ImageNet Data Set[1] contains five domains, i.e., D1 (*ambulance+scooter*), D2 (*taxi+scooter*), D3 (*jeep+scooter*), D4 (*minivan+scooter*) and D5 (*passenger car+scooter*). Data from different domains come from different categories, e.g., *taxi* from D2 and *jeep* from D3, therefore this dataset is

[1]http://www.image-net.org/download-features

Table 2: Description of the ImageNet dataset

	D_1	D_2	D_3	D_4	D_5
#positive instance	1510	1326	1415	1555	986
#negitive instance	1427	1427	1427	1427	1427
#features	1000	1000	1000	1000	1000

proper for transfer learning study. To construct classification problems, we randomly choose two from the five domains, where one is considered as the source domain and the other is considered as the target domain. Therefore, we construct 20 (P_5^2) transfer learning classification problems. Statistics of this dataset is shown in Table 2.

Corel Data Set[2] [Zhuang *et al.*, 2010] includs two different top categories, *flower* and *traffic*. Each top category further consists of four subcategories. We use *flower* as positive instances and *traffic* as negative ones. To construct the transfer learning classification problems, we randomly select one subcategory from *flower* and one from *traffic* as the source domain, and then choose another subcategory of *flower* and another one of *traffic* from the remaining subcategories to construct the target domain. In this way, we can construct 144 ($P_4^2 \cdot P_4^2$) transfer learning classification problems.

Leaves Data Set [Mallah and Orwell, 2013] includes 100 plant species that are divided into 32 different genera, and each specie has 16 instances. We choose four genuses with more than four plant species to construct 4-class classification problems, and use 64 shape descriptor features to represent an instance. Each genus is regarded as a domain. Similar to the construction of ImageNet dataset, we can construct 12 (P_4^2) 4-class classification problems.

4.2 Baseline Methods

We compare our methods with the following baselines,

- The supervised learning algorithm Logistic Regression (LR) [Friedman and Rob, 2010] without transfer learning.
- Transfer component analysis(TCA) [Pan *et al.*, 2011], which aims at learning a low-dimensional representation for transfer learning. Here we also use Logistic Regression as the basic classifier.
- Transfer learning based on stacked autoencoders, the marginalized Stacked Denoising Autoencoders (mSDA) method [Chen *et al.*, 2012].

Implementation Details: After some preliminary experiments, we set $\alpha = 0.5$, $\beta = 0.5$, $\gamma = 0.00001$ and $k = 10$ for the ImageNet and Corel datasets, while $\beta = 0.05$, $k = 5$ and $\gamma = 0.0001$ for the Leaves dataset. For mSDA, we use the authors' source code[3] and adopt the default parameters as reported in [Chen *et al.*, 2012]. For TCA, the number of latent dimensions is carefully tuned, e.g., for the Corel dataset, the number is sampled from [10, 80] with interval 10, and its best results are reported.

[2]http://archive.ics.uci.edu/ml/datasets/Corel+Image+Features.

[3]http://www.cse.wustl.edu/ mchen/

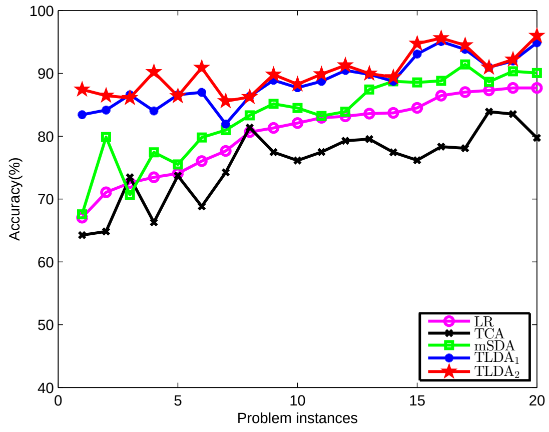


Figure 2: Classification accuracy on the ImageNet dataset

Table 3: Average results (%) on three data sets

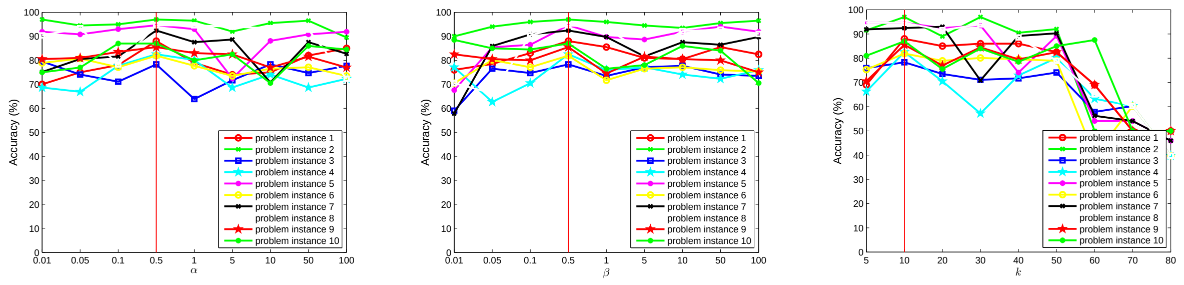
	LR	TCA	mSDA	TLDA$_1$	TLDA$_2$
			ImageNet Data Set		
Left	67.0	64.3	67.6	83.4	**87.4**
Right	81.2	76.3	84.1	89.0	**90.2**
Total	80.5	75.7	83.3	88.7	**90.1**
			Corel Data Set		
Left	61.7	65.4	70.5	71.1	**74.0**
Right	80.1	82.0	75.4	**83.2**	83.0
Total	74.8	76.5	74.0	79.6	**80.4**
			Leaves Data Set		
Left	51.9	**65.9**	47.2	64.1	57.8
Right	75.0	89.8	59.4	**91.4**	89.8
Total	55.7	**69.9**	49.2	68.6	63.2

4.3 Experimental Results

All the results of these three data sets are shown in Figure 2 and Table 3. Figure 2 shows the results over the 20 classification problems on the ImageNet dataset, in which x-axis represents the index of the problems, and y axis represents the corresponding accuracy. From the figure, we have the following observations,

- TLDA is significantly better than LR on every problem, which indicates the efficiency of our proposed transfer learning framework.
- TLDA performs better than TCA, which shows the superiority of applying deep autoencoders to learn a good representation for transfer learning. TLDA also outperforms mSDA, which indicates the effectiveness of incorporating label information from source domain.
- LR performs slightly worse than mSDA, even better than TCA. This may be because on constructed cross-domain classification problems, it is not easy to make knowledge transfer success. This observation again validates the effectiveness of our methods.

We also divide the constructed problems into two groups: a first group consists of problems on which the classification



(a) The Parameter Influence of α (b) The Parameter Influence of β (c) The Parameter Influence of k

Figure 3: The Study of Parameter Influence on TLRA_1

accuracy of LR is lower than 70%, and the rest problems are considered as a second group. The lower of classification accuracy of LR in some certain indicate the higher degree of the difficulty in knowledge transfer. The averaged accuracy of these two group as well as the averaged accuracy over all problems on the three three datasets are reported in Table 3. We can find that the proposed methods perform better than all the compared algorithms on the both groups of problems, except for that on the Leaves dataset, the performance of TLDA_1 is comparable with that of TCA.

4.4 Parameter Sensitivity

In this section, we investigate the influence of the parameters α, β and k in the objective Eq.(7). In this experiment, when tuning one parameter, the values of the rest two are fixed. Specifically, α and β are sampled from $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100\}$, and k is selected from $\{10, 20, ..., 80\}$. We select 10 out of the 144 problems on the Corel dataset for experiment, and report the results in Figure 3. From the figure, we can observe that the performance of TLDA_1 is relatively stable to the selection of α and β, while it decreases dramatically when the value of k is large. Thus we set $\alpha = 0.5$, $\beta = 0.5$ and $k = 10$ to achieve good and stable results for the ImageNet and Corel datasets.

5 Related Work

Poultney *et al.* [2006] proposed an unsupervised method with an energy-based model for learning sparse and overcomplete features. In their method, the decoder produces accurate reconstructions of the patches, while the encoder provides a fast prediction of the code without the need for any particular preprocessing of the inputs. Vincent and Manzagol [2008] proposed Denoising autoencoders to learn a more robust representation from an artificially corrupted input, and further proposed Stacked denoising autoencoders [Vincent *et al.*, 2010] to learn useful representations through a deep network.

Transfer learning has attracted much attention in the past decade. To reduce the difference between domains, two categories of transfer learning approaches have been proposed. One is based on the instance level, which aims to learn weights for the source domain labeled data, such that the reweighted source domain instances look similar to the target domain data instances [Dai *et al.*, 2007b; Gao *et al.*, 2008; Xing *et al.*, 2007; Jiang and Zhai, 2007; Zhuang *et al.*, 2010; Crammer *et al.*, 2012]. The other is based on the feature representation level, which aims to learn a new feature representation for both the source and target domain data, such that with the new feature representation the difference between domains can be reduced [Blitzer *et al.*, 2006; Dai *et al.*, 2007a; Pan *et al.*, 2008; Si *et al.*, 2010; Pan *et al.*, 2011; Xavier and Bengio, 2011; Chen *et al.*, 2012; Zhuang *et al.*, 2014].

Among most feature-based transfer learning methods, only a few methods aim to minimize the difference between domains explicitly in learning the new feature representation. For instance, maximum mean discrepancy embedding (MMDE) [Pan *et al.*, 2008] and transfer component analysis (TCA) [Pan *et al.*, 2011] try to minimize the distance in distributions between domains in a kernel Hilbert space, respectively. The transfer subspace learning framework proposed by [Si *et al.*, 2010] tries to find a subspace, where the distributions of the source and target domain data are similar, through a minimization on the KL divergence of the projected instances between domains. However, they are either based on kernel methods or regularization frameworks, rather than exploring a deep architecture to learn feature representations for transfer learning. Different from previous works, in this paper, our proposed TLDA is a supervised representation learning method based on deep learning, which takes distance minimization between domains and label encoding of the source domain into consideration.

6 Conclusion

In this paper, we proposed a supervised representation learning framework for transfer learning with deep autoencoders. In this framework, the well known representation learning model autoencoder is considered, and we extend it to a deeper architecture. Indeed there are two layers for encoding, one is for embedding, where we impose the KL divergence constrains to draw the two distributions of source and target domains similar. The other is label layer, by which we can easily incorporate the label information from source domain. Finally, we conduct a series of experiments on three real-world image data sets, and all the results demonstrate the effectiveness of the proposed methods.

Acknowledgments.
This work is supported by the National Natural Science Foundation of China (No. 61473273, 61473274, 61175052, 61203297), National High-tech R&D Program of China (863 Program) (No.2014AA015105, 2013AA01A606). Sinno J. Pan is supported by the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020.

References

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[Blitzer *et al.*, 2006] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on EMNLP*, pages 120–128, 2006.

[Chen *et al.*, 2012] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th ICML*, 2012.

[Crammer *et al.*, 2012] Koby Crammer, Mark Dredze, and Fernando Pereira. Confidence-weighted linear classification for text categorization. *The Journal of Machine Learning Research*, 13(1):1891–1926, 2012.

[Dai *et al.*, 2007a] W. Y. Dai, G. R. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD*, 2007.

[Dai *et al.*, 2007b] W. Y. Dai, Q. Yang, G. R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th ICML*, 2007.

[Friedman and Rob, 2010] Hastie Trevor Tibshirani Friedman, Jerome and Rob. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[Gao *et al.*, 2008] J. Gao, W. Fan, J. Jiang, and J. W. Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD*, 2008.

[Jiang and Zhai, 2007] Jing Jiang and Chengxiang Zhai. Instance weighting for domain adaptation in nlp. In *In ACL*, pages 264–271, 2007.

[Joey Tianyi Zhou and Yan, 2014] Ivor Tsang Joey Tianyi Zhou, Sinno Jialin Pan and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *Proceedings of the 28th AAAI*, pages 2213–2220, 2014.

[Kullback, 1987] Solomon Kullback. Letter to the editor: the kullback-leibler distance. 1987.

[Liddle *et al.*, 2010] Andrew R Liddle, Pia Mukherjee, and David Parkinson. Model selection and multi-model inference. *Bayesian Methods in Cosmology*, 1:79, 2010.

[Mallah and Orwell, 2013] Cope Mallah and Orwell. Plant leaf classification using probabilistic integration of shape, texture and margin features. *Signal Processing, Pattern Recognition and Applications*, 2013.

[Pan and Yang, 2010] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transaction on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[Pan *et al.*, 2008] S. J. Pan, J. T. Kwok, and Q. Yang. Transfer learning via dimensionality reduction. In *Proceedings of the 23rd AAAI*, 2008.

[Pan *et al.*, 2011] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, Qiang Yang, et al. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[Poultney *et al.*, 2006] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[Si *et al.*, 2010] Si Si, Dacheng Tao, and Bo Geng. Bregman divergence-based regularization for transfer subspace learning. *IEEE Transaction on Knowledge and Data Engineering*, 22(7):929–942, 2010.

[Snyman, 2005] Jan Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer Science & Business Media, 2005.

[Vincent and Manzagol, 2008] Bengio Vincent, Larochelle and Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[Xavier and Bengio, 2011] Antoine Xavier and Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th ICML*, pages 513–520, 2011.

[Xing *et al.*, 2007] D. K. Xing, W. Y. Dai, G. R. Xue, and Y. Yu. Bridged refinement for transfer learning. In *Proceedings of the 10th PAKDD*. 2007.

[Zhuang *et al.*, 2010] Fuzhen Zhuang, Ping Luo, Hui Xiong, Yuhong Xiong, Qing He, and Zhongzhi Shi. Cross-domain learning from multiple sources: a consensus regularization perspective. *IEEE Transactions on Knowledge and Data Engineering*, 22(12):1664–1678, 2010.

[Zhuang *et al.*, 2014] Fuzhen Zhuang, Xiaohu Cheng, Sinno Jialin Pan, Wenchao Yu, Qing He, and Zhongzhi Shi. Transfer learning with multiple sources via consensus regularized autoencoders. In *Machine Learning and Knowledge Discovery in Databases*, pages 417–431. Springer, 2014.