Neurocomputing ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Domain adaptation via Multi-Layer Transfer Learning

Jianhan Pan [a,*], Xuegang Hu [a], Peipei Li [a], Huizong Li [a], Wei He [a], Yuhong Zhang [a], Yaojin Lin [b]

[a] Hefei University of Technology, Hefei 230009, China
[b] Minnan Normal University, Zhangzhou 363000, China

ARTICLE INFO

Article history:
Received 5 March 2015
Received in revised form
16 December 2015
Accepted 30 December 2015
Communicated by Zhaohong Deng

Keywords:
Transfer Learning
Non-Negative Matrix Tri-Factorization
Multi-Layer
Cross-domain classification

ABSTRACT

Transfer learning, which leverages labeled data in a source domain to train an accurate classifier for classification tasks in a target domain, has attracted extensive research interests recently for its effectiveness proven by many studies. Previous approaches adopt a common strategy that models the shared structure as a bridge across different domains by reducing distribution divergences. However, those approaches totally ignore specific latent spaces, which can be utilized to learn non-shared concepts. Only specific latent spaces contain specific latent factors, lacking which will lead to ineffective distinct concept learning. Additionally, only learning latent factors in one latent feature space layer may ignore those in the other layers. The missing latent factors may also help us to model the latent structure shared as the bridge. This paper proposes a novel transfer learning method Multi-Layer Transfer Learning (MLTL). MLTL first generates specific latent feature spaces. Second, it combines these specific latent feature spaces with common latent feature space into one latent feature space layer. Third, it generates multiple layers to learn the corresponding distributions on different layers with their pluralism simultaneously. Specifically, the pluralism of the distributions on different layers means that learning the distributions on one layer can help us to learn the distributions on the others. Furthermore, an iterative algorithm based on Non-Negative Matrix Tri-Factorization is proposed to solve the optimization problem. Comprehensive experiments demonstrate that MLTL can significantly outperform the state-of-the-art learning methods on topic and sentiment classification tasks.

1. Introduction

Traditional machine learning classification algorithms implicitly assume that the training and test data are drawn from the same distribution. However, this assumption seldom holds in reality. To tackle the challenge of different data distributions, many transfer learning methods have been proposed recently for real-world applications, such as computational biology [11], image classification [12] and text classification [3–7,13,14]. Transfer learning or domain adaptation aims to exploit the labeled examples in the source domain to model a better classifier for predicting the classes of the test examples in the target domain, where there are less or no labeled examples. Some of these previous approaches show that the latent high-level concepts, which are related to feature clusters extracted on the raw features, are more appropriate for the text classification across domains than learning from the original features [7]. In [3], CoCC (Co-clustering based classification for out-of-domain documents) transfers the identical concepts. MTrick [4] exploits the associations between the homogeneous concepts and the example classes as the bridge across domains. DTL (Dual transfer learning) [5] uses the identical and homogeneous concepts as the shared concepts to establish the bridge. In addition, HIDC (Concept Learning for Cross-Domain Text Classification: A General Probabilistic Framework) [7] and Tri-TL [6] exploit the distinct concepts for classification learning besides the shared concepts.

To model the latent structure shared for knowledge transfer, these previous methods usually construct one latent feature space layer to learn the corresponding distributions. We represent such method as the single layer transfer learning. The limitation of these approaches is two-fold:

(1) To build a more effective bridge across domains, some previous methods such as Tri-TL [6] and HIDC [7] learn the shared concepts (including identical and homogeneous concepts) and non-shared concepts (including distinct concepts) simultaneously. Additionally, these approaches construct one common latent feature space and two random latent spaces for

* Corresponding author.
E-mail address: peter.jhpan@gmail.com (J. Pan).

http://dx.doi.org/10.1016/j.neucom.2015.12.097
0925-2312/

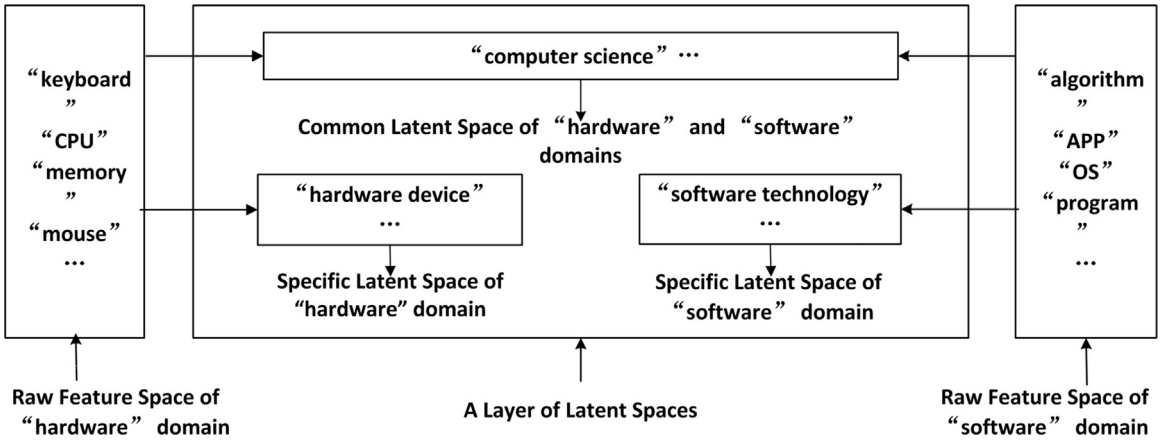


Fig. 1. A latent feature space layer.

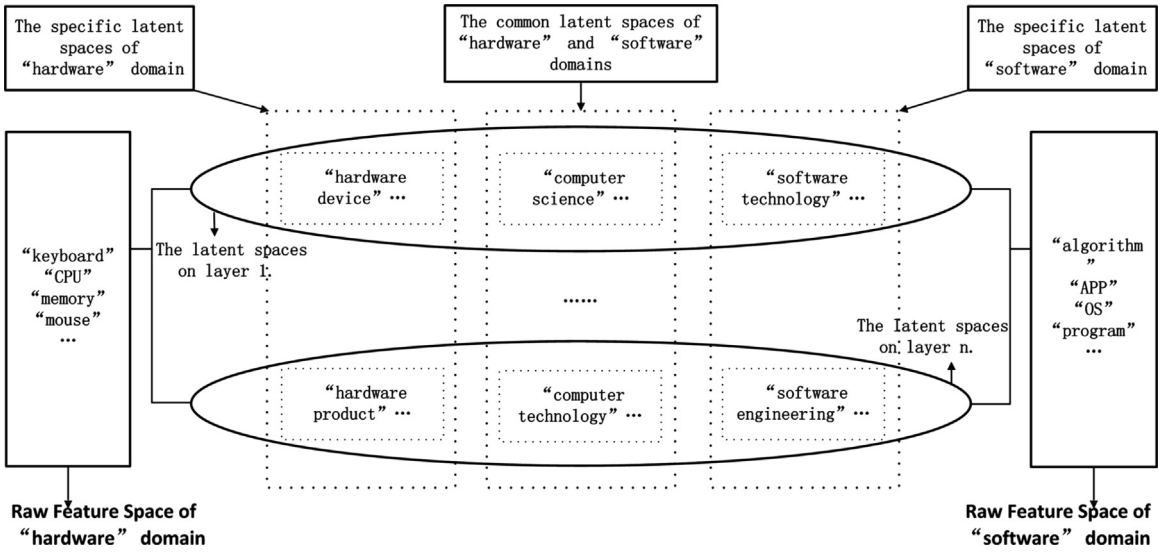


Fig. 2. Learning the distributions on different latent feature space layers.

learning the shared and non-shared concepts respectively (one of the random latent spaces is constructed for learning the distinct concepts in the source domain, and the other one is for the learning in target domain). These latent spaces constitute a latent space layer. The common latent space is composed of feature clusters, which are extracted on the common raw feature space, and the random latent spaces are composed of the random-number clusters. The ideal model should learn the shared concepts on the common latent space, and learn the non-shared concepts on the specific latent spaces. However, these methods did not construct the specific latent spaces for learning non-shared concepts, and using the random latent spaces instead. Actually, the common and random latent space contains few specific latent factors, which are related to the distinct concepts, to discriminate domains.

For example, words like “CPU” and “keyboard”, which are drawn from *hardware* domain, as well as “APP” and “algorithm”, which are drawn from *software* domain, can be indicated to the distinct concepts “hardware device” and “software technology” respectively. These distinct concepts are among the most discriminative latent concepts. In Fig. 1, we can find that these distinct concepts only can be obtained from the corresponding specific latent spaces respectively. Therefore, lack of specific latent factors leads to ineffective distinct concepts learning.

(2) These methods only construct one latent space layer and implicitly assume that all the useful latent factors can be obtained from this latent space layer. However, this assumption seldom holds in reality. The set of the latent factors in one latent space layer is just a subset of all the latent factors. For example, words like “CPU”, “keyboard”, “APP” and “algorithm” can be indicated to the shared concept “computer science”, which exists in a latent space layer. In Fig. 2, we can see that these words can be also indicated to the shared concept “computer technology”, which may exist in another layer. Both of these shared concepts can help us to model the shared structure as the bridge across domain. Therefore, it will ignore some other latent factors to construct the single latent space layer. At worst, when the distribution is dominated by these latent factors and the distribution divergences among domains are so large, this strict assumption will cause negative transfer.

In this paper, we propose Multi-Layer Transfer Learning (MLTL), a novel transfer learning method based on Non-Negative Matrix Tri-Factorization (NMTF) techniques, which constructs specific latent feature spaces and integrates them with the common latent

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

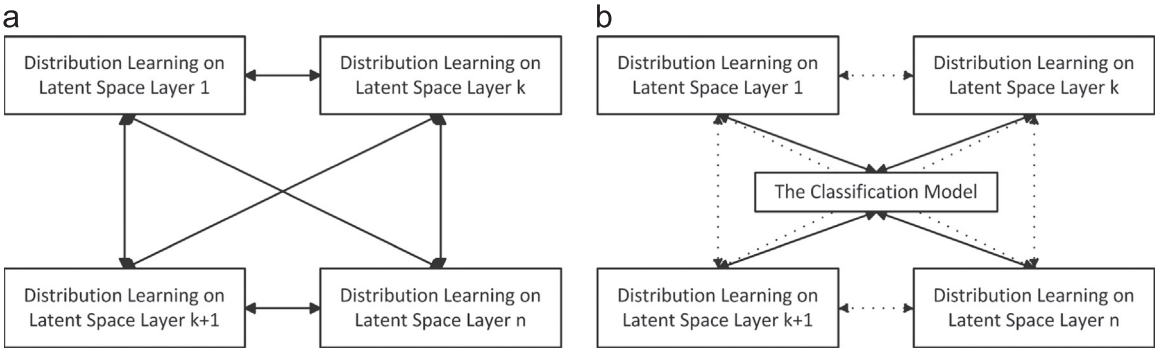


Fig. 3. The pluralism of distributions learning on different latent feature space layers. (a) The direct interaction among the distribution learning on different latent space layers. (b) The indirect interaction among the distribution learning on different latent space layers.

feature space as one latent feature space layer, then constructs multiple layers to learn the corresponding distributions on the different layers simultaneously with the pluralism of them. The key idea of MLTL is as follows. Firstly, as shown in Fig. 1, MLTL constructs the specific latent spaces, and integrates them with the common latent space as one latent feature space layer, then more specific latent factors, which are obtained from the latent spaces in the source and target domains respectively, can be utilized to discriminate domains. Secondly, MLTL constructs multiple latent feature space layers and learns the distributions on different layers simultaneously as shown in Fig. 2. Therefore, we can exploit more latent factors to establish the bridge across domains and make the distributions among domains more closer by using the pluralism of the distributions. The pluralism is that learning the distributions on one layer can help us to learn the distributions on the others. On one hand, learning the distributions on different layers may promote each other directly. On the other hand, they can indirectly facilitate each other as well. Specifically, it exploits the latent factors on one layer to learn the corresponding distributions, and, then, a classification model shared across different layers can be obtained to facilitate the distributions learning on the other layers. In other words, learning the distributions and modeling the classifier can promote each other directly. Therefore, by the classification model shared across different layers, learning the distributions on the different layers can facilitate each other indirectly. We show the interaction among the distributions learning on different layers in Fig. 3.

The main contributions of MLTL are summarized as follows:

(1) MLTL constructs the specific latent feature spaces and integrates them with the common latent feature space as one latent feature space layer. Then more specific latent factors can be utilized to discriminate domains.
(2) MLTL constructs multiple layers to learn the distributions on different layers simultaneously with the pluralism of them. Then more common and specific latent factors can be utilized to establish the bridge across domains for knowledge transfer.
(3) We construct the systematic experiments to show the effectiveness of MLTL. In particular, all the compared methods lead to negative transfer on the traditional tasks and cannot compete with the traditional machine learning method Logistic Regression (LR) on the sentiment tasks. Only MLTL avoid negative transfer on the traditional tasks successfully and obtain the best average accuracies on all the tasks including the sentiment tasks.

The rest of this paper is organized as follows. Section 2 introduces the related work. We review preliminary knowledge in Section 3. In Section 4, we describe the model of MLTL. Section 5 provides the experimental results. Finally, Section 6 concludes our work in this paper.

2. Related works

In this section, we discuss previous works that are most related to our work. According to literature survey [1,27], most previous methods can be divided into two categories including the weighting-based methods and the feature representation-based methods.

Weighting-based approaches can be further divided into two subcategories: the instance weighting-based and model weighting-based methods. Instance weighting-based methods focus on a re-weighting strategy [19,25]. The general idea for these methods is to increase the weight of the data in source domain which is close to the data in the target domain; otherwise, decrease the weight. From the view of Instance weighting, Jiang et al. [25] proposed a general framework based on instance weighting to deal with NLP tasks. Dai et al. [19] extended a boosting-style learning algorithm for cross-domain learning by the re-weighting strategy. On the other hand, the model weighting-based methods give the classification model different weights. Gao et al. [20] proposed a dynamic model weight-based method according to the similarity between the local structure in the target domain and the classification model.

Our work belongs to the feature representation-based methods, which can also be grouped into two subcategories further, including the feature selection-based and feature mapping-based approaches. The general strategy of feature selection-based approaches is to select the general features which are useful for domain adaptation from the raw features among different domains [3,21,22]. Dai et al. [3] proposed a coclustering-based method, which identifies the word clusters among the source and target domains, by propagating the class information and knowledge from the source domain to the target domain. Jiang et al. [21] developed a two-step feature selection framework for knowledge transfer with the strategy that the features highly related to class labels should be assigned to large weights in the learnt model. Uguroglu et al. [22] proposed an approach to identify variant and invariant features between two data sets for transfer learning. On the other side, feature mapping-based approaches, to which MLTL belongs, map the raw feature space to a latent high-level feature

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Table 1
Notations and descriptions.

Notations	Descriptions
$\mathcal{D}_r$	Domain r
r	Domain index
s	The number of source domains
t	The number of target domains
X_r	The feature-example co-occurrence matrix of $\mathcal{D}_r$
U	The matrix of feature clusters
H	The matrix of the association between feature clusters and example classes
V	The matrix of example classes
m	The number of original features
n_r	The number of examples in domain $\mathcal{D}_r$
c	The number of example classes
e	Layer index $1 \le e \le NOL$
l_e	Layer e
NOL	The number of layers
T	Transposition of matrix

space, under a guidance of the principle that the source and the target domains are drawn from the same distribution [2,4–7,17,23,24,28]. Blitzer et al. [2] presented a correspondence learning method, which first identifies the correspondence among features and then explores this correspondence for knowledge transfer. Zhuang et al. [4] proposed a method to exploit the associations between feature clusters and example clusters for cross-domain learning. Long et al. [5] proposed a method, called Dual Transfer Learning (DTL). It utilizes the duality between the marginal and conditional distributions. Chen and Zhang [17] proposed TCL to construct a latent space, and, then learn the common and specific words to discriminate domains. In [6], the proposed Triplex Transfer Learning (Tri-TL) based on nonnegative matrix tri-factorization (NMTF) analyzes the three kinds of concepts (identical, homogeneous and distinct concepts), then models them together. The proposed method HIDC [7] is similar to Tri-TL but considering the distribution of domains. Pan et al. [23] proposed an algorithm to find out the latent feature space based on dimensionality reduction. Long et al. [24] proposed a method based on property preservation, which extracts shared latent factors between domains by preserving the important geometric structure properties of the original data.

Additionally, some other transfer learning methods also refer to the latent space [28,30–36]. Shell et al. [28] proposed a framework, Fuzzy Transfer Learning (FuzzyTL), which combines the Fuzzy Logic (FL) and Transfer Learning (TL) to transfer knowledge. Zheng et al. [30] proposed a latent multi-task learning algorithm to solve the multi-device indoor localization problem. Ji et al. [31] presented a general framework for extracting shared structures in multi-label classification. Jiang et al. [32] proposed a general framework for preserving the independent information among different tasks and mining hidden correlation information among all tasks in multi-task fuzzy modeling. Pan et al. [33] developed a novel framework of Transfer by Collective Factorization (TCF), in which they construct a shared latent space collectively and learn the data-dependent effect separately. Deng et al. [34] presented a fuzzy system with knowledge-leverage capability, which is known as a knowledge-leverage-based fuzzy system (KLFS). In [35], a knowledge-leverage-based Takagi–Sugeno–Kang-type Fuzzy System (KL-TSK-FS) is proposed by integrating the corresponding knowledge-leverage mechanism. In [36], the generalized hidden-mapping ridge regression (GHRR) method is introduced in order to train various types of classical intelligence models, including neural networks, fuzzy logical systems and kernel methods.

These transfer learning approaches which construct one latent space layer can be represented as single layer methods. The limitations of these single layer methods are two-fold: (1) these single layer methods including TCL, Tri-TL, and HIDC ignored that one latent feature space layer should include not only the common latent space but also the specific latent spaces, which contain more specific latent factors to discriminate domains. (2) Some latent factors, which may also help us to model the shared structure, may exist on the other latent feature space layers. Along this line, the Multi-Layer Transfer Learning (MLTL) method is proposed. This model integrates the common and specific latent spaces as one latent feature space layer, and, then, constructs multiple layers to learn the distributions on different layers simultaneously with the pluralism of them. Therefore, more latent factors can be utilized to establish the bridge across domains. Moreover, we demonstrate the effectiveness of MLTL compared with the existing approaches by extensive experimental evaluation.

3. Preliminary knowledge

In this section, we first give some basic concepts and mathematical notations used in this paper, and, then introduce the Non-Negative Matrix Tri-Factorization (NMTF) model briefly.

3.1. Notations

In this paper, we represent data matrix with uppercase, such as X and Y, and denote the element at the i-th row and j-th column of matrix X as $X_{[i,j]}$. Then, we use 1_m to represent a vector with m rows and one column, and each element in this vector is equal to 1. Then, the set of real numbers and non-negative real numbers is denoted as R and R_+ respectively. Let $\mathcal{D}=(\mathcal{D}_1,\cdots,\mathcal{D}_s,\mathcal{D}_{s+1},\cdots,\mathcal{D}_{s+t})$ be a family of data domains which has s source domains and t target domains. The first s domains are source domains denoted as $\mathcal{D}_r=\{x_i^r,y_i^r\}|_{i=1}^{n_r}(1\le r\le s)$, where y_i^r is the label of the example x_i^r, and the rest t domains are target domains denoted as $\mathcal{D}_r=\{x_i^r\}|_{i=1}^{n_r}(s+1\le r\le s+t)$ that the documents are unlabeled. The variables r and n_r represent the index of domains and the number of examples in the domain $\mathcal{D}_r$ respectively. For each domain $\mathcal{D}_r(1\le r\le s+t)$, we represent the feature-example co-occurrence matrix as $X_r\in R_+^{m\times n_r}$ under the assumption that all the elements of the matrix X_r which involves m features and n_r examples are non-negative. In Table 1, we summarize the frequently used notations in MLTL.

3.2. Non-Negative Matrix Tri-Factorization

Since the nonnegative matrix tri-factorization (NMTF) model has been widely used for text data classification [3-7,10,13,14], we briefly introduce NMTF, on which our model is based. In NMTF, the matrix $X\in R^{m\times n}$ approximates a product of three nonnegative factors $U\in R^{m\times k}, H\in R^{k\times c}$, and $V\in R^{n\times c}$, such that $X\approx UHV^T$. Then we can obtain the basic formula as follows:

$$X_{m\times n}=U_{m\times k}H_{k\times c}V_{n\times c}^T \tag{1}$$

where $X\in R^{m\times n}$ represents the feature-example matrix. m, n, k, and c represents the numbers of raw features, examples, high-level feature clusters, and example classes respectively. $U\in R^{m\times k}$ represents a feature-cluster matrix owning k feature clusters and m raw features which actually means a mapping that from raw features to high-level features. The high-level feature clusters can be seen to the probability distributions over m raw features. $V\in R^{n\times c}$ represents an example-class matrix owning c classes and n examples which refers to the mapping that from examples to classes. The class clusters in V can be described as the probability

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Table 2
Three kinds of concepts.

Notation	Description
c_1	**Identical concepts**: When the source and target domains have the same concept extension and the same concept intension, these high-level concepts are identical
c_2	**Homogeneous concepts**: When the source and target domains have the different concept extension and the same concept intension, these high-level concepts are homogeneous
c_3	**Distinct concepts**: When the source and target domains have the different concept extension and the different concept intension, these high-level concepts are distinct

distributions over n examples. $H \in R^{k\times c}$ represents a cluster-class matrix owning k high-level feature clusters and c classes which means the association between high-level feature clusters and examples classes. The class clusters in H can be viewed as the probability distributions over k high-level feature clusters.

Additionally, NMTF which can be solved by an iterative update algorithm is an optimization problem, and we show the formula as follows:

$$\min_{U,H,V\geq 0} \quad \|X-UHV^T\|^2$$
$$\text{s.t.} \quad \sum_{i=1}^{m} U_{[i,j]}=1, \quad \sum_{j=1}^{c} V_{[i,j]}=1 \tag{2}$$

To adapt transfer learning, NMTF is extended to multiple domains. The formula of the NMTF optimization problem can be rewritten as

$$\min_{U,H,V_r\geq 0} \quad \sum_{r=1}^{s+t} \|X_r-U_rHV_r^T\|^2$$
$$\text{s.t.} \quad \sum_{i=1}^{m} U_{r[i,j]}=1, \quad \sum_{j=1}^{c} V_{r[i,j]}=1 \tag{3}$$

Here, we also introduce some basic concepts about NMTF used in Section 4

Definition 3.1 (*Trace of matrix*). Given a data matrix $X \in R^{n\times n}$, the trace of X is computed as

$$\mathrm{Tr}(X)=\sum_{i=1}^{n} X_{[ii]} \tag{4}$$

In fact, we can compute the trace of matrix square when the matrix is a square matrix.

Definition 3.2 (*Frobenius norm of matrix*). Given a data matrix $X \in R^{m\times n}$, the Frobenius norm of X is computed as

$$\|X\|^2=\sum_{i=1}^{m}\sum_{j=1}^{n} X_{[ij]}^2 \tag{5}$$

Next, we list some conversion formulas of the trace of matrix.

Property 3.1. *Given a matrix* $X \in R^{m\times n}$, *then*

$$\mathrm{Tr}\left(X^TX\right)=\mathrm{Tr}\left(XX^T\right) \tag{6}$$

Property 3.2. *Given two matrices* $X, Y \in R^{m\times n}$, *then*

$$\mathrm{Tr}(a\cdot X+b\cdot Y)=a\cdot \mathrm{Tr}(X)+b\cdot \mathrm{Tr}(Y) \tag{7}$$

Property 3.3. *Given a matrix* $X \in R^{m\times n}$, *then*

$$\|X\|^2=\mathrm{Tr}\left(X^TX\right)=\mathrm{Tr}\left(XX^T\right) \tag{8}$$

4. Multi-Layer Transfer Learning

In this section, we first define the problem setting and formulate the MLTL model as an optimization problem. After that, we introduce the solution to the model and the proof of algorithm convergence. Finally, we analyze the computational complexity of the algorithm.

4.1. Problem definition

Since the latent spaces are represented by the corresponding high-level concepts or feature clusters, we systemically analyze the high-level concepts, and then lay out two kinds of explanations of high-level concepts with different perspectives as follows: The set of high-level concepts are represented as *Concepts Extension*, and the association between these high-level concepts and the example classes is represented as *Concepts Intension* [6,7]. On one latent feature space layer, when the source and target domains have the same concept extension and the same concept intension, we denote these concepts as the identical concepts. When the source and target domains have the different concept extension and the same concept intension, we denote them as the homogeneous concepts. Both of the identical and homogeneous concepts are related to the common latent space. Similarly, when the source and target domains have the different concept extension and the different concept intension, we denote them as the distinct concepts, which are related to the specific latent spaces. The description of these high-level concepts on one latent feature space layer is shown in Table 2.

To fit different data distribution situations on one latent space layer, MLTL learns the identical, homogeneous and distinct concepts together. Along this line, we divide U and H into three parts. Specifically, $U=\left[U^{c_1}_{m\times k_{c_1}}, U^{c_2}_{m\times k_{c_2}}, U^{c_3}_{m\times k_{c_3}}\right](k_{c_1}+k_{c_2}+k_{c_3}=k)$, and $U^{c_1}_{m\times k_{c_1}}$ represents the feature clusters for the identical concepts, $U^{c_2}_{m\times k_{c_2}}$ represents the feature clusters for the homogeneous concepts and $U^{c_3}_{m\times k_{c_3}}$ represents the feature clusters for the distinct concepts. Accordingly, H can be indicated as

$$H=\begin{bmatrix} H^{c_1}_{k_{c_1}\times c} \\ H^{c_2}_{k_{c_2}\times c} \\ H^{c_3}_{k_{c_3}\times c} \end{bmatrix},$$

and $H^{c_1}_{k_{c_1}\times c}$ represents the association between example classes and the identical concepts, $H^{c_2}_{k_{c_2}\times c}$ represents the association between example classes and the homogeneous concepts, and $H^{c_3}_{k_{c_3}\times c}$ represents the association between example classes and the

distinct concepts. Therefore, formula (1) can be rewritten as

$$X_{m\times n}=U_{m\times k}H_{k\times c}V^T_{n\times c}=\left[U^{c_1}_{m\times k_{c_1}},U^{c_2}_{m\times k_{c_2}},U^{c_3}_{m\times k_{c_3}}\right]\begin{bmatrix}H^{c_1}_{k_{c_1}\times c}\\H^{c_2}_{k_{c_2}\times c}\\H^{c_3}_{k_{c_3}\times c}\end{bmatrix}V^T_{n\times c} \quad (9)$$

Then, the objective function is formulated as follows:

$$\mathcal{L}=\sum_{r=1}^{s+t}\|X_r-U_rH_rV_r^T\|^2 \quad (10)$$

where $X_r\in R_+^{m\times n}, U_r\in R_+^{m\times k}, H_r\in R_+^{k\times c}$ and $V_r^T\in R_+^{n^r\times c}$.

According to formula (9), the objective function can be rewritten as follows:

$$\mathcal{L}=\sum_{r=1}^{s+t}\|X_r-U_rH_rV_r^T\|^2=\sum_{r=1}^{s+t}\|X_r-\left[U^{c_1},U_r^{c_2},U_r^{c_3}\right]\begin{bmatrix}H^{c_1}\\H^{c_2}\\H_r^{c_3}\end{bmatrix}V_r^T\|^2 \quad (11)$$

For Eq. (11), $U^{c_1}\in R_+^{m\times k_{c_1}}$ represents the set of feature clusters for the identical concepts. $U_r^{c_2}\in R_+^{m\times k_{c_2}}$ represents the set of feature clusters for the homogeneous concepts. $U_r^{c_3}\in R_+^{m\times k_{c_3}}$ represents the set of feature clusters for the distinct concepts. $H^{c_1}\in R_+^{k_{c_1}\times c}$ represents the association between the identical concepts and example classes. $H^{c_2}\in R_+^{k_{c_2}\times c}$ represents the association between the homogeneous concepts and example classes. $H_r^{c_3}\in R_+^{k_{c_3}\times c}$ represents the association between the distinct concepts and example classes. $V_r\in R_+^{n_r\times c}(s+1\le r\le s+t)$ represents the classifier used in target domain.

To utilize more latent factors for training a more effective classification model, we first construct multiple different latent space layers by running a clustering algorithm with different parameters respectively, and then simultaneously learn the corresponding distributions on the different latent feature space layers with the pluralism of them as the bridge across domains. In this paper, to describe the effectiveness of MLTL in brief, the number of latent space layers is set to three $e\in[1,3]$. Then we can obtain the corresponding feature clusters related to learning the distributions on the different latent space layers, and the sets of these feature clusters which are represented as U_1^c, $U_r^{c_2}$ and $U_r^{c_3}$ can be divided into three parts according to these different latent feature space layers respectively, such that $U^{c_1}=[U^{c_1}_{l_1},U^{c_1}_{l_2},U^{c_1}_{l_3}]$, where $U^{c_1}\in R_+^{m\times k_{c_1}}, U^{c_1}_{l_1}\in R_+^{m\times k^{c_1}_{l_1}}, U^{c_1}_{l_2}\in R_+^{m\times k^{c_1}_{l_2}}, U^{c_1}_{l_3}\in R_+^{m\times k^{c_1}_{l_3}}$ and $k^{c_1}_{l_1}+k^{c_1}_{l_2}+k^{c_1}_{l_3}=k_{c_1}$. $U_r^{c_2}=[U^{c_2}_{r(l_1)},U^{c_2}_{r(l_2)},U^{c_2}_{r(l_3)}]$, where $U_r^{c_2}\in R_+^{m\times k_{c_2}}, U^{c_2}_{r(l_1)}\in R_+^{m\times k^{c_2}_{l_1}}, U^{c_2}_{r(l_2)}\in R_+^{m\times k^{c_2}_{l_2}}, U^{c_2}_{r(l_3)}\in R_+^{m\times k^{c_2}_{l_3}}$ and $k^{c_2}_{l_1}+k^{c_2}_{l_2}+k^{c_2}_{l_3}=k_{c_2}$. $U_r^{c_3}=[U^{c_3}_{r(l_1)},U^{c_3}_{r(l_2)},U^{c_3}_{r(l_3)}]$, where $U_r^{c_3}\in R_+^{m\times k_{c_3}}, U^{c_3}_{r(l_1)}\in R_+^{m\times k^{c_3}_{l_1}}, U^{c_3}_{r(l_2)}\in R_+^{m\times k^{c_3}_{l_2}}, U^{c_3}_{r(l_3)}\in R_+^{m\times k^{c_3}_{l_3}}$ and $k^{c_3}_{l_1}+k^{c_3}_{l_2}+k^{c_3}_{l_3}=k_{c_3}$. Similarly, H^{c_1}, H^{c_2} and $H_r^{c_3}$, which are represented as the association between example classes and the identical concepts, the association between example classes and the homogeneous concepts and the association between example classes and the distinct concepts respectively, can also be divided into three parts according to the different latent feature space layers respectively, such that $H^{c_1}=[H^{c_1}_{l_1},H^{c_1}_{l_2},H^{c_1}_{l_3}]$, where $H^{c_1}\in R_+^{k_{c_1}\times c}, H^{c_1}_{l_1}\in R_+^{k^{c_1}_{l_1}\times c}, H^{c_1}_{l_2}\in R_+^{k^{c_1}_{l_2}\times c}, H^{c_1}_{l_3}\in R_+^{k^{c_1}_{l_3}\times c}$ and $k^{c_1}_{l_1}+k^{c_1}_{l_2}+k^{c_1}_{l_3}=k_{c_1}$. $H^{c_2}=[H^{c_2}_{l_1},H^{c_2}_{l_2},H^{c_2}_{l_3}]$, where $H^{c_2}\in R_+^{k_{c_2}\times c}, H^{c_2}_{l_1}\in R_+^{k^{c_2}_{l_1}\times c}, H^{c_2}_{l_2}\in R_+^{k^{c_2}_{l_2}\times c}, H^{c_2}_{l_3}\in R_+^{k^{c_2}_{l_3}\times c}$ and $k^{c_2}_{l_1}+k^{c_2}_{l_2}+k^{c_2}_{l_3}=k_{c_2}$. $H_r^{c_3}=[H^{c_3}_{r(l_1)},H^{c_3}_{r(l_2)},H^{c_3}_{r(l_3)}]$, where $H_r^{c_3}\in R_+^{k_{c_3}\times c}, H^{c_3}_{r(l_1)}\in R_+^{k^{c_3}_{l_1}\times c}, H^{c_3}_{r(l_2)}\in R_+^{k^{c_3}_{l_2}\times c}, H^{c_3}_{r(l_3)}\in R_+^{k^{c_3}_{l_3}\times c}$ and $k^{c_3}_{l_1}+k^{c_3}_{l_2}+k^{c_3}_{l_3}=k_{c_3}$. Then we model these distributions on different layers simultaneously by taking advantage of the pluralism of them. Then we can rewrite Eq. (11) as follows:

$$\mathcal{L}=\sum_{r=1}^{s+t}\|X_r-U_rH_rV_r^T\|^2$$
$$=\sum_{r=1}^{s+t}\|X_r-\left[U^{c_1}_{l_1},U^{c_1}_{l_2},U^{c_1}_{l_3},U^{c_2}_{r(l_1)},U^{c_2}_{r(l_2)},U^{c_2}_{r(l_3)}U^{c_3}_{r(l_1)},U^{c_3}_{r(l_2)},U^{c_3}_{r(l_3)}\right]\begin{bmatrix}H^{c_1}_{l_1}\\H^{c_1}_{l_2}\\H^{c_1}_{l_3}\\H^{c_2}_{l_1}\\H^{c_2}_{l_2}\\H^{c_2}_{l_3}\\H^{c_3}_{r(l_1)}\\H^{c_3}_{r(l_2)}\\H^{c_3}_{r(l_3)}\end{bmatrix}V_r^T\|^2 \quad (12)$$

To quantify the relationships among the original features, the feature clusters and the example classes, we model the constraint conditions for $U^{c_1}_{l_1},U^{c_1}_{l_2},U^{c_1}_{l_3},U^{c_2}_{r(l_1)},U^{c_2}_{r(l_2)},U^{c_2}_{r(l_3)},U^{c_3}_{r(l_1)},U^{c_3}_{r(l_2)},U^{c_3}_{r(l_3)},H^{c_1}_{l_1},H^{c_1}_{l_2},H^{c_1}_{l_3},H^{c_2}_{l_1},H^{c_2}_{l_2},H^{c_2}_{l_3},H^{c_3}_{r(l_1)},H^{c_3}_{r(l_2)},H^{c_3}_{r(l_3)}$ and V_r simultaneously, and educe the optimization problem as follows:

$$\begin{aligned}&\min_{U_r,H_r,V_r}\ \mathcal{L}\\&\text{s.t.}\ \sum_{j=1}^{k^{c_1}_{l_1}}U^{c_1}_{(l_1)[i,j]}=1,\ \sum_{j=1}^{k^{c_1}_{l_2}}U^{c_1}_{(l_2)[i,j]}=1,\ \sum_{j=1}^{k^{c_1}_{l_3}}U^{c_1}_{(l_3)[i,j]}=1,\\&\sum_{j=1}^{k^{c_2}_{l_1}}U^{c_2}_{r(l_1)[i,j]}=1,\ \sum_{j=1}^{k^{c_2}_{l_2}}U^{c_2}_{r(l_2)[i,j]}=1,\ \sum_{j=1}^{k^{c_2}_{l_3}}U^{c_2}_{r(l_3)[i,j]}=1,\\&\sum_{j=1}^{k^{c_3}_{l_1}}U^{c_3}_{r(l_1)[i,j]}=1,\ \sum_{j=1}^{k^{c_3}_{l_2}}U^{c_3}_{r(l_2)[i,j]}=1,\ \sum_{j=1}^{k^{c_3}_{l_3}}U^{c_3}_{r(l_3)[i,j]}=1,\\&\sum_{j=1}^{c}H^{c_1}_{(l_1)[i,j]}=1,\ \sum_{j=1}^{c}H^{c_1}_{(l_2)[i,j]}=1,\ \sum_{j=1}^{c}H^{c_1}_{(l_3)[i,j]}=1,\\&\sum_{j=1}^{c}H^{c_2}_{(l_1)[i,j]}=1,\ \sum_{j=1}^{c}H^{c_2}_{(l_2)[i,j]}=1,\ \sum_{j=1}^{c}H^{c_2}_{(l_3)[i,j]}=1,\\&\sum_{j=1}^{c}H^{c_3}_{r(l_1)[i,j]}=1,\ \sum_{j=1}^{c}H^{c_3}_{r(l_2)[i,j]}=1,\ \sum_{j=1}^{c}H^{c_3}_{r(l_3)[i,j]}=1,\sum_{j=1}^{c}V_{r[i,j]}=1.\end{aligned} \quad (13)$$

Here, these constraint conditions represent the feature cluster distribution of the original features, the class distribution of the feature clusters, and the class distribution of the examples respectively.

4.2. MLTL solution

To solve the method MLTL, we first give the analysis of the objective function, and, then derive the updating rules. Additionally an iterative algorithm is proposed to solve the optimization problem. The formulation of the objective function is

$$\mathcal{L}=\sum_{r=1}^{s+t}\|X_r-\left[U^{c_1}_{l_e},U^{c_2}_{r(l_e)},U^{c_3}_{r(l_e)}\right]\begin{bmatrix}H^{c_1}_{l_e}\\H^{c_2}_{l_e}\\H^{c_3}_{r(l_e)}\end{bmatrix}V_r^T\|^2$$
$$=\sum_{r=1}^{s+t}\mathrm{Tr}\left(X_r^TX_r-2\cdot X_r^T\left[U^{c_1}_{l_e},U^{c_2}_{r(l_e)},U^{c_3}_{r(l_e)}\right]\begin{bmatrix}H^{c_1}_{l_e}\\H^{c_2}_{l_e}\\H^{c_3}_{r(l_e)}\end{bmatrix}V_r^T+V_r\begin{bmatrix}H^{c_1}_{l_e}\\H^{c_2}_{l_e}\\H^{c_3}_{r(l_e)}\end{bmatrix}^T\right.$$

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Table 3
Descriptions of the number of concepts.

Number of concepts on layers \ Number of different types of concepts	The number of identical concepts	The number of homogeneous concepts	The number of distinct concepts	Total number of concepts
The number of concepts on layer 1	$k_{l_1}^{c_1}$	$k_{l_1}^{c_2}$	$k_{l_1}^{c_3}$	k_{l_1}
The number of concepts on layer 2	$k_{l_2}^{c_1}$	$k_{l_2}^{c_2}$	$k_{l_2}^{c_3}$	k_{l_2}
The number of concepts on layer 3	$k_{l_3}^{c_1}$	$k_{l_3}^{c_2}$	$k_{l_3}^{c_3}$	k_{l_3}
Total number of concepts	k_{c_1}	k_{c_2}	k_{c_3}	k

$$\times\left[U_{l_e}^{c_1}, U_{r(l_e)}^{c_2}, U_{r(l_e)}^{c_3}\right]^T\left[U_{l_e}^{c_1}, U_{r(l_e)}^{c_2}, U_{r(l_e)}^{c_3}\right]\begin{bmatrix} H_{l_e}^{c_1} \\ H_{l_e}^{c_2} \\ H_{r(l_e)}^{c_3}\end{bmatrix} V_r^T\Bigg)$$

$$= \sum_{r=1}^{s+t} \mathrm{Tr}\Big(X_r^T X_r - 2\cdot X_r^T A_r - 2\cdot X_r^T B_r - 2\cdot X_r^T C_r + A_r^T A_r + B_r^T B_r + C_r^T C_r + 2\cdot A_r^T B_r + 2\cdot A_r^T C_r + 2\cdot B_r^T C_r\Big)$$

$$\text{s.t.}\quad \sum_{j=1}^{k_{c_1}} U_{[i,j]l_e}^{c_1} = 1,\quad \sum_{j=1}^{k_{c_2}} U_{r[i,j]l_e}^{c_2} = 1,\quad \sum_{j=1}^{k_{c_3}} U_{r[i,j]l_e}^{c_3} = 1,$$

$$\sum_{j=1}^{c} H_{[i,j]l_e}^{c_1} = 1,\quad \sum_{j=1}^{c} H_{[i,j]l_e}^{c_2} = 1,\quad \sum_{j=1}^{c} H_{r[i,j]l_e}^{c_3} = 1,\quad \sum_{j=1}^{c} V_{r[i,j]} = 1. \tag{14}$$

where $A_r = U_{l_e}^{c_1} H_{l_e}^{c_1} V_r^T, A_r^T = V_r H_{l_e}^{c_1^T} U_{l_e}^{c_1^T}, B_r = U_{r(l_e)}^{c_2} H_{l_e}^{c_2} V_r^T, B_r^T = V_r H_{l_e}^{c_2^T} U_{r(l_e)}^{c_2^T}, C_r = U_{r(l_e)}^{c_3} H_{r(l_e)}^{c_3} V_r^T, C_r^T = V_r H_{r(l_e)}^{c_3^T} U_{r(l_e)}^{c_3^T}$ and $e \in [1,3]$.

Since the objective function is not concave, it is unrealistic to get a global solution by using the non-linear optimization technique. Then we propose an alternately iterative algorithm with convergence guarantee to obtain a local optimal solution. The reason for adopting iterative algorithm is that iterative algorithm is easy to implement and its convergence properties are guaranteed. Other algorithms are possibly more efficient in overall computation time, but are more difficult to implement and cannot generalize to different cost functions [15]. The comparison between the iterative approach and others is outside the scope of this paper. Since all the compared transfer methods in this paper adopt iterative approach, we only focus on the strategy for knowledge transfer. These variables are updated as:

$$U_{[i,j]l_e}^{c_1} \leftarrow U_{[i,j]l_e}^{c_1} \cdot \left(\left[\sum_{r=1}^{s+t} X_r V_r H_{l_e}^{c_1}\hat{T}\right]_{[i,j]} / \left[\sum_{r=1}^{s+t} A_r V_r H_{l_e}^{c_1}\hat{T} + B_r V_r H_{l_e}^{c_1}\hat{T} + C_r V_r H_{l_e}^{c_1}\hat{T}\right]_{[i,j]}\right)^{1/2} \tag{15}$$

$$U_{r[i,j]l_e}^{c_2} \leftarrow U_{r[i,j]l_e}^{c_2} \cdot ([X_r V_r H_{l_e}^{c_2}\hat{T}]_{[i,j]} / [A_r V_r H_{l_e}^{c_2}\hat{T} + B_r V_r H_{l_e}^{c_2}\hat{T} + C_r V_r H_{l_e}^{c_2}\hat{T}]_{[i,j]})^{1/2} \tag{16}$$

$$U_{r[i,j]l_e}^{c_3} \leftarrow U_{r[i,j]l_e}^{c_3} \cdot ([X_r V_r H_{r(l_e)}^{c_3}\hat{T}]_{[i,j]} / [A_r V_r H_{r(l_e)}^{c_3}\hat{T} + B_r V_r H_{r(l_e)}^{c_3}\hat{T} + C_r V_r H_{r(l_e)}^{c_3}\hat{T}]_{[i,j]})^{1/2} \tag{17}$$

$$H_{[i,j]l_e}^{c_1} \leftarrow H_{[i,j]l_e}^{c_1} \cdot \left(\left[\sum_{r=1}^{s+t} U_{l_e}^{c_1}\hat{T} X_r V_r\right]_{[i,j]} / \left[\sum_{r=1}^{s+t} (U_{l_e}^{c_1}\hat{T} A_r V_r + U_{l_e}^{c_1}\hat{T} B_r V_r + U_{l_e}^{c_1}\hat{T} C_r V_r)\right]_{[i,j]}\right)^{1/2} \tag{18}$$

$$H_{[i,j]l_e}^{c_2} \leftarrow H_{[i,j]l_e}^{c_2} \cdot \left(\left[\sum_{r=1}^{s+t} U_{r(l_e)}^{c_2} T X_r V_r\right]_{[i,j]} / \left[\sum_{r=1}^{s+t} (U_{r(l_e)}^{c_2} T A_r V_r + U_{r(l_e)}^{c_2} T B_r V_r + U_{r(l_e)}^{c_2} T C_r V_r)\right]_{[i,j]}\right)^{1/2} \tag{19}$$

$$H_{r[i,j]l_e}^{c_3} \leftarrow H_{r[i,j]l_e}^{c_3} \cdot \left(\left[U_{r(l_e)}^{c_3}\hat{T} X_r V_r\right]_{[i,j]} / \left[U_{r(l_e)}^{c_3}\hat{T} A_r V_r + U_{r(l_e)}^{c_3}\hat{T} B_r V_r + U_{r(l_e)}^{c_3}\hat{T} C_r V_r\right]_{[i,j]}\right)^{1/2} \tag{20}$$

$$V_{r[i,j]} \leftarrow V_{r[i,j]} \cdot \left(\left[X_r^T U_r H_r\right]_{[i,j]} / \left[V_r H_r^T U_r^T U_r H_r\right]_{[i,j]}\right)^{1/2} \tag{21}$$

In each time of iteration, we calculate all the variables under the updating rules and use Eq. (22) to normalize $U_{l_e}^{c_1}, U_{r(l_e)}^{c_2}, U_{r(l_e)}^{c_3}$, $H_{l_e}^{c_1}, H_{l_e}^{c_2}, H_{r(l_e)}^{c_3}$ and V_r:

$$U_{[i,j]l_e}^{c_1} \leftarrow \frac{U_{[i,j]l_e}^{c_1}}{\sum_{j=1}^{k_{c_1}} U_{[i,j]l_e}^{c_1}}, U_{r[i,j]l_e}^{c_2} \leftarrow \frac{U_{r[i,j]l_e}^{c_2}}{\sum_{j=1}^{k_{c_2}} U_{r[i,j]l_e}^{c_2}}, U_{r[i,j]l_e}^{c_3} \leftarrow \frac{U_{r[i,j]l_e}^{c_3}}{\sum_{j=1}^{k_{c_3}} U_{r[i,j]l_e}^{c_3}},$$
$$H_{[i,j]l_e}^{c_1} \leftarrow \frac{H_{[i,j]l_e}^{c_1}}{\sum_{j=1}^{c} H_{[i,j]l_e}^{c_1}}, H_{[i,j]l_e}^{c_2} \leftarrow \frac{H_{[i,j]l_e}^{c_2}}{\sum_{j=1}^{c} H_{[i,j]l_e}^{c_2}}, H_{r[i,j]l_e}^{c_3} \leftarrow \frac{H_{r[i,j]l_e}^{c_3}}{\sum_{j=1}^{c} H_{r[i,j]l_e}^{c_3}},$$
$$V_{r[i,j]} \leftarrow \frac{V_{r[i,j]}}{\sum_{j=1}^{c} V_{r[i,j]}} \tag{22}$$

Based on the above formulas, we propose an iterative algorithm and describe it in Algorithm 1.[1] In this algorithm, $V_r (1 \le r \le s)$ is initialized by the true label information, and $V_r (1+s \le r \le s+t)$ is initialized by Logistic Regression (LR) [8] which is trained on the source domain to make the algorithm converge faster. We normalize the data matrices such that $X_r^T 1_m = 1_n$. $U_{l_1}^{c_1}, U_{l_2}^{c_1}, U_{l_3}^{c_1}, U_{r(l_1)}^{c_2}$, $U_{r(l_2)}^{c_2}, U_{r(l_3)}^{c_2}, U_{r(l_1)}^{c_3}, U_{r(l_2)}^{c_3}$ and $U_{r(l_3)}^{c_3}$ are initialized with the feature clusters which are obtained by implemented PLSA [9]. According to the number setting of the feature clusters showed in Table 3, we can obtain the feature information $W_{l_1}^{c_1} \in R_+^{m\times k_{l_1}^{c_1}}, W_{l_2}^{c_1} \in R_+^{m\times k_{l_2}^{c_1}}, W_{l_3}^{c_1} \in R_+^{m\times k_{l_3}^{c_1}}, W_{l_1}^{c_2} \in R_+^{m\times k_{l_1}^{c_2}}, W_{l_2}^{c_2} \in R_+^{m\times k_{l_2}^{c_2}}$ and $W_{l_3}^{c_2} \in R_+^{m\times k_{l_3}^{c_2}}$ in the corresponding latent space layers through conducting PLSA on the data from the source and target domains, and obtain $W_{r(l_1)}^{c_3} \in R_+^{m\times k_{l_1}^{c_3}}$, $W_{r(l_2)}^{c_3} \in R_+^{m\times k_{l_2}^{c_3}}$ and $W_{r(l_3)}^{c_3} \in R_+^{m\times k_{l_3}^{c_3}}$ $(1 \le r \le s+t)$ in the corresponding latent space layers through conducting PLSA on the data only from one domain $\mathcal{D}_r (1 \le r \le s+t)$. Then $U_{l_1}^{c_1}, U_{l_2}^{c_1}, U_{l_3}^{c_1}, U_{r(l_1)}^{c_2}, U_{r(l_2)}^{c_2}, U_{r(l_3)}^{c_2}$, $U_{r(l_1)}^{c_3}, U_{r(l_2)}^{c_3}$ and $U_{r(l_3)}^{c_3}$ are initialized as $W_{l_1}^{c_1}, W_{l_2}^{c_1}, W_{l_3}^{c_1}, W_{l_1}^{c_2}, W_{l_2}^{c_2}, W_{l_3}^{c_2}$, $W_{r(l_1)}^{c_3}, W_{r(l_2)}^{c_3}$ and $W_{r(l_3)}^{c_3}$ respectively.

[1] The source codes can be downloaded from https://sourceforge.net/projects/translearn/files/latest/download

Algorithm 1. MLTL: Multi-Layer Transfer Learning Algorithm.

Input: $\{X_r\}_{r=1}^{s+t}$, $\{V_r\}_{r=1}^{s}$, parameters $k_{l_1}^{c_1}, k_{l_2}^{c_1}, k_{l_3}^{c_1}, k_{l_1}^{c_2}, k_{l_2}^{c_2}, k_{l_3}^{c_2}, k_{l_1}^{c_3}, k_{l_2}^{c_3}, k_{l_3}^{c_3}$ and the number of iterations *maxIter*.

Output: $U_{l_1}^{c_1}, U_{l_2}^{c_1}, U_{l_3}^{c_1}, U_{r(l_1)}^{c_2}, U_{r(l_2)}^{c_2}, U_{r(l_3)}^{c_2}, U_{r(l_1)}^{c_3}, U_{r(l_2)}^{c_3}, U_{r(l_3)}^{c_3}, H_{l_1}^{c_1}, H_{l_2}^{c_1}, H_{l_3}^{c_1}, H_{l_1}^{c_2}, H_{l_2}^{c_2}, H_{l_3}^{c_2}, H_{r(l_1)}^{c_3}, H_{r(l_2)}^{c_3}, H_{r(l_3)}^{c_3}$ $(1 \le r \le s+t)$ and $V_r (1+s \le r \le s+t)$.

Normalize the data matrices by $X_{r[i,j]} \leftarrow X_{r[i,j]} / \sum_{i=1}^{m} X_{r[i,j]}$, $(1 \le r \le s+t)$;

The initializations of $U_{l_1}^{c_1(0)}, U_{l_2}^{c_1(0)}, U_{l_3}^{c_1(0)}, U_{r(l_1)}^{c_2(0)}, U_{r(l_2)}^{c_2(0)}, U_{r(l_3)}^{c_2(0)}, U_{r(l_1)}^{c_3(0)}, U_{r(l_2)}^{c_3(0)}$ and $U_{r(l_3)}^{c_3(0)}$ are detailed in section 4.2, $H_{l_1}^{c_1(0)}, H_{l_2}^{c_1(0)}, H_{l_3}^{c_1(0)}, H_{l_1}^{c_2(0)}, H_{l_2}^{c_2(0)}, H_{l_3}^{c_2(0)}, H_{r(l_1)}^{c_3(0)}, H_{r(l_2)}^{c_3(0)}$ and $H_{r(l_3)}^{c_3(0)}$ are randomly assigned, and $V_r^{(0)} (1+s \le r \le s+t)$ is initialized by Logistic Regression;

for $k \leftarrow 1$ *to maxIter* **do**

  Update $U_{l_1}^{c_1(k)}, U_{l_2}^{c_1(k)}$ and $U_{l_3}^{c_1(k)}$, by Eq.(15);

  for $r \leftarrow 1$ *to* $s+t$ **do**

    Update $U_{r(l_1)}^{c_2(k)}, U_{r(l_2)}^{c_2(k)}$ and $U_{r(l_3)}^{c_2(k)}$ by Eq.(16);

  end

  for $r \leftarrow 1$ *to* $s+t$ **do**

    Update $U_{r(l_1)}^{c_3(k)}, U_{r(l_2)}^{c_3(k)}$ and $U_{r(l_3)}^{c_3(k)}$ by Eq.(17);

  end

  Update $H_{l_1}^{c_1(k)}, H_{l_2}^{c_1(k)}$ and $H_{l_3}^{c_1(k)}$ by Eq.(18);

  Update $H_{l_1}^{c_2(k)}, H_{l_2}^{c_2(k)}$ and $H_{l_3}^{c_2(k)}$ by Eq.(19);

  for $r \leftarrow 1$ *to* $s+t$ **do**

    Update $H_{r(l_1)}^{c_3(k)}, H_{r(l_2)}^{c_3(k)}$ and $H_{r(l_3)}^{c_3(k)}$ by Eq.(20);

  end

  for $r \leftarrow s+1$ *to* $s+t$ **do**

    Update $V_r^{(k)}$ by (21);

  end

  Normalize $U_{l_1}^{c_1(k)}, U_{l_2}^{c_1(k)}, U_{l_3}^{c_1(k)}, U_{r(l_1)}^{c_2(k)}, U_{r(l_2)}^{c_2(k)}, U_{r(l_3)}^{c_2(k)}, U_{r(l_1)}^{c_3(k)}, U_{r(l_2)}^{c_3(k)}, U_{r(l_3)}^{c_3(k)}, H_{l_1}^{c_1(k)}, H_{l_2}^{c_1(k)}, H_{l_3}^{c_1(k)}, H_{l_1}^{c_2(k)}, H_{l_2}^{c_2(k)}, H_{l_3}^{c_2(k)}, H_{r(l_1)}^{c_3(k)}, H_{r(l_2)}^{c_3(k)}, H_{r(l_3)}^{c_3(k)}$ $(1 \le r \le s+t)$ and $V_r^{(k)} (1+s \le r \le s+t)$ by (22);

  Output $U_{l_1}^{c_1(k)}, U_{l_2}^{c_1(k)}, U_{l_3}^{c_1(k)}, U_{r(l_1)}^{c_2(k)}, U_{r(l_2)}^{c_2(k)}, U_{r(l_3)}^{c_2(k)}, U_{r(l_1)}^{c_3(k)}, U_{r(l_2)}^{c_3(k)}, U_{r(l_3)}^{c_3(k)}, H_{l_1}^{c_1(k)}, H_{l_2}^{c_1(k)}, H_{l_3}^{c_1(k)}, H_{l_1}^{c_2(k)}, H_{l_2}^{c_2(k)}, H_{l_3}^{c_2(k)}, H_{r(l_1)}^{c_3(k)}, H_{r(l_2)}^{c_3(k)}, H_{r(l_3)}^{c_3(k)}$ $(1 \le r \le s+t)$ and $V_r^{(k)} (1+s \le r \le s+t)$.

end

4.3. Theoretical analysis

In this section, we formulate the Lagrange function of the objective function with constraint to prove the convergence of updating rules (15)–(21), which were derived following the theory of constraint optimization [29]:

$$\mathcal{L} = \sum_{r=1}^{s+t} \|X_r - U_r H_r V_r^T\| + \sum_{r=1}^{s+t} \mathrm{Tr}(\Gamma_r (U_r 1_k - 1_m)(U_r 1_k - 1_m)^T) + \sum_{r=1}^{s+t} \mathrm{Tr}(\lambda_r (H_r 1_c - 1_k)(H_r 1_c - 1_k)^T) + \sum_{r=1}^{s+t} \mathrm{Tr}(\Lambda_r (V_r 1_c - 1_{n_r})(V_r 1_c - 1_{n_r})^T) \quad (23)$$

where $\Gamma_r \in R_+^{m \times m}, \lambda_r \in R_+^{k \times k}$ and $\Lambda_r \in R_+^{n_r \times n_r}$ are diagonal matrixes and represent the Lagrange multipliers.

Since the derivation process of the updating rule for different variables is similar, we only display the detailed derivation process for V_r without losing generality. Therefore, Eq. (23) becomes

$$\mathcal{L}(V_r) = \sum_{r=1}^{s+t} \mathrm{Tr}\left(-2 \cdot X_r^T U_r H_r V_r^T + V_r H_r^T U_r^T U_r H_r V_r^T\right) + \sum_{r=1}^{s+t} \mathrm{Tr}(\Lambda_r (V_r 1_c - 1_{n_r})(V_r 1_c - 1_{n_r})^T) \quad (24)$$

The differential is

$$\frac{\partial \mathcal{L}}{\partial V_r} = -2 \cdot X_r^T U_r H_r + 2 \cdot V_r H_r^T U_r^T U_r H_r + 2 \Lambda_r V_r 1_c 1_c^T - 2 \Lambda_r 1_{n_r} 1_c^T \quad (25)$$

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Table 4
The top categories and their subcategories.

Top categories	Subcategories
comp	*comp.graphics, sys.mac.hardware* *comp.sys.ibm.pc.hardware* *comp.os.ms-windows.misc*
rec	*rec.autos, motorcycles* *rec.sport.baseball, hockey*
sci	*sci.crypt, med, electronics, space*
talk	*talk.politics.guns, Mideast, misc* *talk.religion.misc*

The update formula is as follows:

$$V_{r[i,j]} \leftarrow V_{r[i,j]} \cdot \sqrt{\frac{\left[X_r^T U_r H_r\right]_{[i,j]} + \Lambda_r 1_{n_r} 1_c^T}{\left[V_r H_r^T U_r^T U_r H_r\right]_{[i,j]} + \Lambda_r V_r 1_c 1_c^T}} \tag{26}$$

Lemma 4.1. *Eq. (24) is monotone decreasing while using the updating rule (26).*

To prove Lemma 4.1, we provide the definitions of auxiliary function [15] as follows:

Definition 4.1. $\mathcal{J}(Y, \tilde{Y})$ is an auxiliary function of $\mathcal{L}(Y)$ if it satisfies

$$\mathcal{J}(Z, \tilde{Z}) \geq \mathcal{L}(Z) \quad \text{and} \quad \mathcal{J}(Z, Z) = \mathcal{L}(Z) \tag{27}$$

for any $Z, \tilde{Z}$.

Definition 4.2. $\mathcal{J}(Y, \tilde{Y})$ is an auxiliary function of $\mathcal{L}(Y)$ if it satisfies the following formula:

$$Z^{(t+1)} = \underset{Z}{\operatorname{argmin}}\ \mathcal{J}(Z, Z^{(t)}) \tag{28}$$

Through these definitions, we derive Eq. (29) as follows:

$$\mathcal{L}(Z^{(t)}) = \mathcal{J}\left(Z^{(t)}, Z^{(t)}\right) \geq \mathcal{J}\left(Z^{(t+1)}, Z^{(t)}\right) \geq \mathcal{J}\left(Z^{(t+1)}, Z^{(t+1)}\right) = \mathcal{L}\left(Z^{(t+1)}\right) \tag{29}$$

From Eq. (29), we can find that $\mathcal{L}(Z)$ is monotone decreasing while utilizing the updating rule of Z which is derived by minimizing the auxiliary function $\mathcal{J}(Z, \tilde{Z})$. Then the auxiliary function for $\mathcal{L}(V_r)$ is constructed as

$$\begin{aligned}\mathcal{J}\left(V_r, \tilde{V}_r\right) = &\sum_{i,j} (\tilde{V}_r H_r^T U_r^T U_r H_r + \Lambda_r \tilde{V}_r 1_c 1_c^T)_{[i,j]} \frac{(V_r)_{[i,j]}^2}{\left(\tilde{V}_r\right)_{[i,j]}} \\ &- 2\sum_{i,j} (X_r^T U_r H_r + \Lambda_r 1_{n_r} 1_c^T)_{[i,j]} (\tilde{V}_r)_{[i,j]} \left(1 + \log \frac{(V_r)_{[i,j]}}{(\tilde{V}_r)_{[i,j]}}\right)\end{aligned} \tag{30}$$

Apparently, Equality $\mathcal{L}(V_r) = \mathcal{J}\left(V_r, \tilde{V}_r\right)$ holds when $V_r = \tilde{V}_r$. We can also prove $\mathcal{J}(Z, \tilde{Z}) \geq \mathcal{L}(Z)$ and the Hessian matrix $\nabla\nabla_{V_r}\mathcal{J}\left(V_r, \tilde{V}_r\right) \succcurlyeq 0$ by the similar approaches in [16]. Then we minimize $\mathcal{J}\left(V_r, \tilde{V}_r\right)$ when the variable $\tilde{V}_r$ is fixed. The differential of $\mathcal{J}\left(V_r, \tilde{V}_r\right)$ is

$$\begin{aligned}\frac{\partial \mathcal{J}\left(V_r, \tilde{V}_r\right)}{\partial V_{r[i,j]}} = &2\sum_{i,j} (\tilde{V}_r H_r^T U_r^T U_r H_r + \Lambda_r \tilde{V}_r 1_c 1_c^T)_{[i,j]} \frac{(V_r)_{[i,j]}}{(\tilde{V}_r)_{[i,j]}} \\ &- 2\sum_{i,j} (X_r^T U_r H_r + \Lambda_r 1_{n_r} 1_c^T)_{[i,j]} \frac{(\tilde{V}_r)_{[i,j]}}{(V_r)_{[i,j]}}\end{aligned} \tag{31}$$

Let $\frac{\partial \mathcal{J}(V_r, \tilde{V}_r)}{\partial V_{r[i,j]}} = 0$, we achieve the updating rule (26) which can decrease the value of $\mathcal{L}(V_r)$. Then Lemma 4.1 holds.

To satisfy the constraint conditions, we can use an iterative normalization technique from [14]. Specifically, we normalize V_r by Eq. (22) in each iteration so that $V_r 1_c = 1_{n_r}$. Then we get the equation $\Lambda_r 1_{n_r} 1_c^T = \Lambda_r V_r 1_c 1_c^T$ which depends on Λ_r only. Therefore, the influence of Eqs. (21) and (22) can be approximately updating rule of (26) without influencing convergence. Then we adopt the updating equation (21) for V_r by omitting the equal items which only depend on Λ_r.

Theorem 4.1. *The objective function in (13) will not increase in Algorithm 1 at each iteration.*

Following the similar method mentioned above, we could verify the convergence of the update rules for the rest variables respectively. Thus, Algorithm 1 will not increase (13), and Theorem 4.1 holds. Since the objective function is lower bounded by zero, the convergence of Algorithm 1 is proved.

4.4. Computational complexity of the iterative algorithm

In this section, we analyze the computational complexity of the iterative algorithm. For each round of iteration in Algorithm 1, the computational complexity to calculate $U_{l_1}^{c_1}$ is $O(\sum_{r=1}^{s+t} 19mn_r c + 10mck_{l_1}^{c_1} + mkc + mk_{l_1}^{c_1})$. Since $k \ll n$ and $k_{l_1}^{c_1} \ll n$, the computational complexity can be rewritten as $\mathcal{O}(\sum_{r=1}^{s+t} mn_r c)$. Similarly, the computational complexity of the formulas to calculate $U_{l_2}^{c_1}, U_{l_3}^{c_1}, U_{r(l_1)}^{c_2}, U_{r(l_2)}^{c_2}, U_{r(l_3)}^{c_2}, U_{r(l_1)}^{c_3}, U_{r(l_2)}^{c_3}, U_{r(l_3)}^{c_3}, H_{l_1}^{c_1}, H_{l_2}^{c_1}, H_{l_3}^{c_1}, H_{l_1}^{c_2}, H_{l_2}^{c_2}, H_{l_3}^{c_2}, H_{r(l_1)}^{c_3}, H_{r(l_2)}^{c_3}, H_{r(l_3)}^{c_3}$ and V_r are $\mathcal{O}(\sum_{r=1}^{s+t} mn_r c)$, $\mathcal{O}(\sum_{r=1}^{s+t} mn_r c)$, $\mathcal{O}(mn_r c)$, $\mathcal{O}(mn_r c)$, $\mathcal{O}(mn_r c)$, $\mathcal{O}(mn_r c)$, $\mathcal{O}(mn_r c)$, $\mathcal{O}(mn_r c)$, $\mathcal{O}(\sum_{r=1}^{s+t} mk_{l_1}^{c_1} n_r)$, $\mathcal{O}(\sum_{r=1}^{s+t} mk_{l_2}^{c_1} n_r)$, $\mathcal{O}(\sum_{r=1}^{s+t} mk_{l_3}^{c_1} n_r)$, $\mathcal{O}(\sum_{r=1}^{s+t} mk_{l_1}^{c_2} n_r)$, $\mathcal{O}(\sum_{r=1}^{s+t} mk_{l_2}^{c_2} n_r)$, $\mathcal{O}(\sum_{r=1}^{s+t} mk_{l_3}^{c_2} n_r)$, $\mathcal{O}(mk_{l_1}^{c_3} n_r)$, $\mathcal{O}(mk_{l_2}^{c_3} n_r)$, $\mathcal{O}(mk_{l_3}^{c_3} n_r)$ and $\mathcal{O}(mkn_r)$ respectively. Generally, $c \ll k$, so the maximal computational intensity in each round of iteration is $\mathcal{O}(\sum_{r=1}^{s+t} mkn_r)$. In summary, the computational complexity of Algorithm 1 is $\mathcal{O}(\sum_{r=1}^{s+t} maxIter \cdot mkn_r)$.

5. Experimental evaluation

In this section, we select 20-Newsgroups and Sentiment Data as benchmark datasets, then compare MLTL with other popular supervised, semi-supervised, and domain adaptation approaches.

5.1. Data preparation

The datasets for classification are generated from 20-Newsgroups and Sentiment [26]. First of all, we demonstrate the effectiveness of MLTL on topic classification tasks with 20-Newsgroups dataset. Additionally, we also show the adaptability of MLTL on multi-source sentiment classification tasks with Sentiment dataset.

20-*Newsgroups*[2] includes approximately 20,000 newsgroup examples evenly distributed in 20 different news-groups [1,18–20]. Some of newsgroups are similar and can be classified into a top category, e.g., the four subcategories *sci.crypt*, *sci.med*, *sci.electronics* and *sci.space* belong to the top category *sci*. The details of these top categories and subcategories are demonstrated in Table 4. To verify the validity of MLTL, we construct the tasks as follows:

On one hand, we select two top categories *rec* and *sci* as positive class and negative class respectively. To produce the source domain, we randomly choose two subcategories from *rec* and *sci* respectively. The target domain is constructed similarly. Thus, 144

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

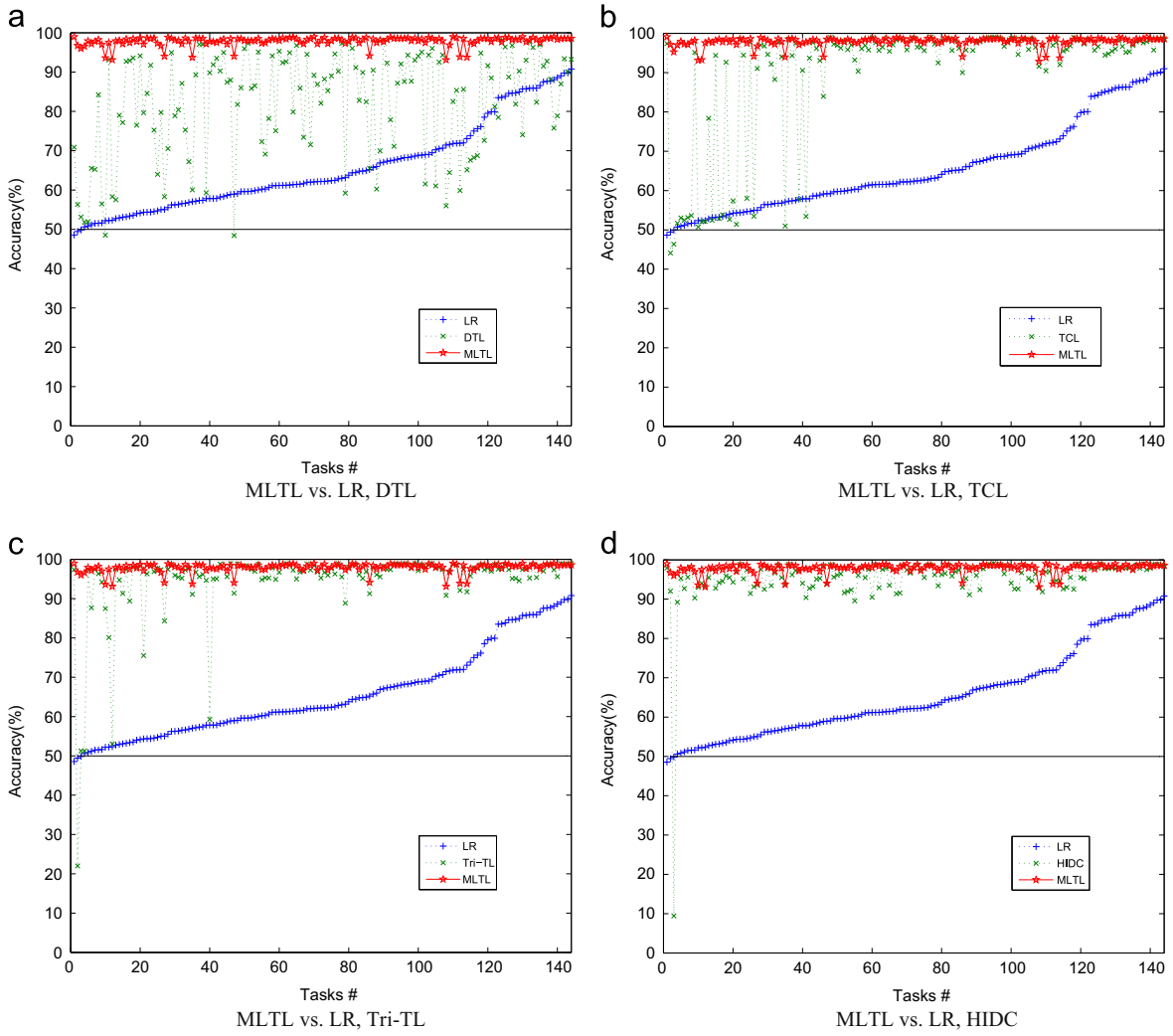


Fig. 4. The performance comparison among LR, DTL, TCL, Tri-TL, HIDC and MLTL on 144 traditional tasks. (a) MLTL vs. LR, DTL. (b) MLTL vs. LR, TCL. (c) MLTL vs. LR, Tri-TL. (d) MLTL vs. LR, HIDC.

Table 5
Performances (%) on 144 traditional tasks.

Algorithms	LR	NMTF	DTL	TCL	Tri-TL	HIDC	MLTL
Average accuracy of 53 tasks lower than 60%	55.21	67.96	77.87	79.58	90.54	93.34	**97.49**
Average accuracy of 91 tasks higher than 60%	71.60	72.54	84.74	96.89	97.05	96.35	**98.07**
Average accuracy of total 144 tasks	65.57	70.86	82.23	89.92	94.65	95.25	**97.85**
Num of negative transfer	–	–	21	13	1	1	**0**

Table 6
Performances (%) on 334 new tasks.

Algorithms	LR	NMTF	DTL	TCL	Tri-TL	HIDC	MLTL
Average accuracy of 126 tasks lower than 60%	54.85	73.4	81.26	81.65	85.8	92.58	**96.02**
Average accuracy of 208 tasks higher than 60%	73.72	80.93	89.97	90.93	95.85	96.62	**98.19**
Average accuracy of total 334 tasks	66.6	78.09	86.69	87.23	92.06	95.09	**97.37**
Num of negative transfer	–	–	31	32	13	4	**2**

$(P_4^2 \times P_4^2)$ tasks are produced in this way. We call these tasks as traditional classification tasks.

On the other hand, for the traditional classification tasks, we replace one subcategory which belongs to *sci* as another subcategory which is from the top categories *comp* and *talk* in the target domain. Then, 384 $(P_4^2 \times P_4^1 \times 8)$ tasks are produced. In this new kind of tasks, the target domain includes another top category which does not exist in the source domain. Therefore, more specific factors exist in the source and the target domains. To avoid negative transfer, only 334 tasks are selected that the initial accuracies of these tasks set by Logistic Regression (LR) [8] are higher than 50%. We call these more difficult tasks as the new tasks. In summary, we have 144 traditional tasks and 334 new tasks form 20-Newsgroups dataset.

Sentiment data[3] includes the positive reviews and the negative reviews from *books*, *dvd*, *electronics* and *kitchen* domains. To demonstrate the adaptability of MLTL on multi-source tasks, we generate the tasks with two source domains selected randomly from the four domains, and one target domain selected from the

[3] http://www.cs.jhu.edu/mdredze/datasets/sentiment

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Table 7
Performances (%) on sentiment tasks.

Algorithms	LR	NMTF	DTL	TCL	Tri-TL	HIDC	MLTL
Average accuracy (400 examples in each domain)	74.23	58.75	72.0	58.26	58.27	71.31	**76.4**
Average accuracy (1000 examples in each domain)	76.35	59.24	76.82	60.24	58.78	76.68	**77.82**
Average accuracy (2000 examples in each domain)	80.28	59.89	77.48	59.45	59.24	**81.5**	80.32
Average accuracy of Total 36 tasks	76.95	59.29	75.4	59.32	58.76	76.5	**78.18**
Num of negative transfer	–	–	29	36	36	22	**14**

Table 8
Parameter setting of the single layer method MLTL-L1.

MLTL-L1	The number of identical concepts	The number of homogeneous concepts	The number of distinct concepts
Number of concepts on layer 1	9	9	4
Number of concepts on layer 2	0	0	0
Number of concepts on layer 3	0	0	0

Table 9
Parameter setting of the single layer method MLTL-L2.

MLTL-L2	The number of identical concepts	The number of homogeneous concepts	The number of distinct concepts
Number of concepts on layer 1	0	0	0
Number of concepts on layer 2	10	10	5
Number of concepts on layer 3	0	0	0

Table 10
Parameter setting of the single layer method MLTL-L3.

MLTL-L3	The number of identical concepts	The number of homogeneous concepts	The number of distinct concepts
Number of concepts on layer 1	0	0	0
Number of concepts on layer 2	0	0	0
Number of concepts on layer 3	11	11	6

Table 11
Performances (%) comparison between the single layer methods and MLTL.

Algorithms	MLTL-L1	MLTL-L2	MLTL-L3	MLTL
Traditional tasks				
Average performances	96.97	97.39	96.58	**97.85**
Num of negative transfer	**0**	**0**	1	**0**
New tasks				
Average performances	96.53	96.49	97.01	**97.37**
Num of negative transfer	4	4	3	**2**
Sentiment tasks				
Average performances	73.32	73.13	73.42	**78.18**
Num of negative transfer	27	30	29	**14**

rest two domains. Then, 12 tasks are generated. To show that MLTL can be applied to the sentiment data set with different examples, we select 400, 1000 and 2000 examples in each domain respectively. Then 36 sentiment tasks are produced.

Table 12
Performances (%) comparison between the single common space methods and the single layer methods.

Algorithms	L1-S	MLTL-L1	L2-S	MLTL-L2	L3-S	MLTL-L3
Traditional tasks						
Average performances	95.13	**96.97**	95.25	**97.39**	94.87	**96.58**
Num of negative transfer	**0**	**0**	**0**	**0**	**1**	**1**
New tasks						
Average performances	94.09	**96.53**	94.61	**96.49**	94.75	**97.01**
Num of negative transfer	10	**4**	7	**4**	6	**3**
Sentiment tasks						
Average performances	73.19	**73.32**	72.54	**73.13**	72.96	**73.42**
Num of negative transfer	30	**27**	31	**30**	31	**29**

Table 13
Performances (%) comparison among the multi-layer methods.

Algorithms	MLTL2	MLTL	MLTL4
Traditional tasks			
Average performances	97.58	97.85	**97.9**
Num of negative transfer	**0**	**0**	**0**
New tasks			
Average performances	97.07	**97.37**	97.31
Num of negative transfer	3	**2**	**2**
Sentiment tasks			
Average performances	76.67	78.18	**78.25**
Num of negative transfer	18	**14**	13

5.2. Experimental setting

Compared algorithms: We compare several state-of-the-art methods with MLTL in the experiments: (1) Supervised method, including Logistic Regression (LR). Since our model MLTL and the compared methods all belong to local optimization methods, which require an initial guess for the optimization variable, and initialize the optimization variable, which refers to the probabilities that a document belongs to different document classes, as the probabilistic output by LR, to obtain a fair result, we adopt results of LR as the initial value and adopt LR as the baseline. (2) Semi-supervised method, including Non-Negative Matrix Tri-Factorization (NMTF). This baseline is trained on both of the source and target domains. Since our model MLTL and most of the compared methods are all based on NMTF, we adopt this model as another baseline. (3) Transfer learning methods, including Dual Transfer Learning (DTL) [5], TCL [17], Tri-TL [6] and HIDC [7]. They are trained on all data and tested on the target domain data. Additionally, the focus of these transfer methods and MLTL is within the semantic domain.

Parameter setting: In MLTL, we set $k_{l_1}^{c_1}=9, k_{l_1}^{c_2}=9, k_{l_1}^{c_3}=4, k_{l_2}^{c_1}=10, k_{l_2}^{c_2}=10, k_{l_2}^{c_3}=5,$ $k_{l_3}^{c_1}=11, k_{l_3}^{c_2}=11, k_{l_3}^{c_3}=6,$ and

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Table 14
The parameter influence on performance (%) of Algorithm MLTL.

Sampling ID	$k_{l_1}^{c_1}$	$k_{l_1}^{c_2}$	$k_{l_1}^{c_3}$	$k_{l_2}^{c_1}$	$k_{l_2}^{c_2}$	$k_{l_2}^{c_3}$	$k_{l_3}^{c_1}$	$k_{l_3}^{c_2}$	$k_{l_3}^{c_3}$	Problem ID						
										1	2	3	4	5	6	7
1	5	8	2	13	11	6	10	12	8	97.27	97.83	98.38	98.18	98.38	97.98	98.84
2	12	7	5	6	9	3	15	13	4	97.02	97.78	98.38	98.13	98.53	97.87	98.64
3	9	11	4	14	8	6	10	7	5	96.96	97.83	98.48	98.08	98.48	98.18	98.84
4	7	12	5	10	13	7	9	11	7	96.92	97.83	98.43	98.13	98.33	98.08	98.84
5	11	9	6	15	14	10	12	10	6	97.07	97.83	98.03	98.13	98.48	98.08	98.89
6	14	13	7	8	6	2	8	8	3	97.22	98.08	98.38	98.08	98.38	97.87	98.59
7	6	7	1	11	9	3	6	7	4	97.02	97.73	98.48	97.92	98.89	98.53	98.79
8	8	10	4	12	14	7	13	9	6	97.07	98.03	98.38	98.08	98.69	98.08	98.89
9	10	12	9	6	10	4	10	11	7	97.27	97.98	98.58	98.08	98.23	97.92	98.89
Mean										97.09	97.88	98.39	98.09	98.49	98.07	98.8
Variance										0.017	0.015	0.024	0.005	0.039	0.042	0.013
This paper	9	9	4	10	10	5	11	11	6	97.07	97.98	98.28	98.13	98.53	98.03	98.84

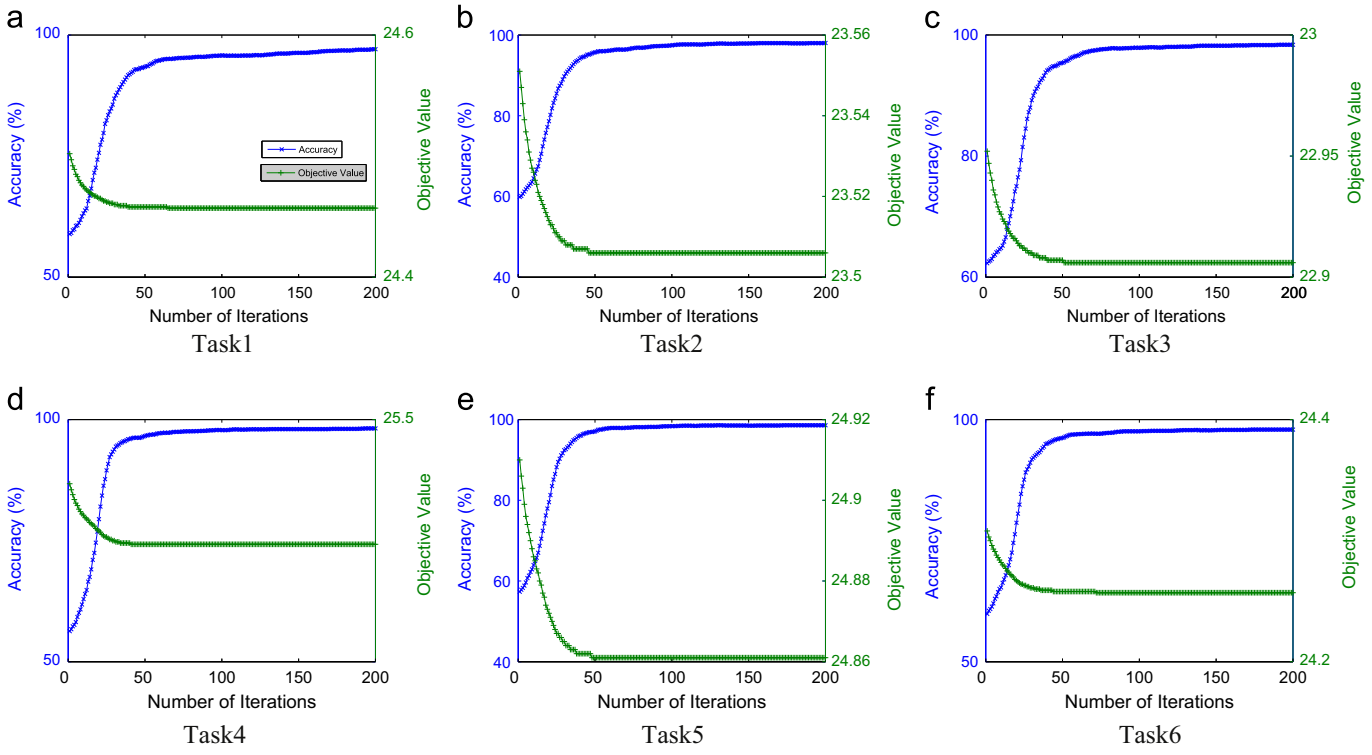


Fig. 5. The performance of MLTL and objective value vs. the number of iterations.

maxIter=200. The baseline method LR is implemented by Matlab[4], NMTF is given by [17]. The parameters of DTL, TCL, Tri-TL and HIDC are set as the default ones in their papers.

We utilize the classification accuracy which is widely used as the evaluation metric,

$$Accuracy = \frac{|\{d : d \in D \wedge f(d) = y(d)\}|}{n}$$

where $y(d)$ is the true label of example d, $f(d)$ is the label predicted by the classification model and n is the number of the examples.

5.3. Experimental results

In this section, we compare our MLTL with LR, NMTF, DTL, TCL, Tri-TL and HIDC on the traditional tasks, the new tasks and the sentiment tasks, and show the results in Fig. 4, and Tables 5–7. In Tables 5 and 6, "Tasks lower than 60%" and "Tasks higher than 60%" represent the tasks, on which LR outputs the accuracy lower than 60% and higher than 60% respectively. In 144 traditional tasks, the accuracy of LR is lower than 60% on 53 tasks while it is higher than 60% on the remaining 91 tasks. In 334 new tasks, the accuracy of LR is lower than 60% on 126 tasks while it is higher than 60% on the remaining 208 tasks. In Fig. 4, we sort the traditional tasks according to the ascending order of the performance of LR and utilize a horizontal line to distinguish whether negative transfer occurs. Negative transfer happens when the source domain data and task contribute to the reduced performance of learning in the target domain [1]. In this paper, we consider that a transfer learning method occurs negative transfer on a classification task when the accuracy is lower than the probability of random selection or lower than the accuracy of a traditional machine learning method. Specifically, since we only focus on binary text classification in the experiments, when the accuracy is lower than 50%, which is below the horizontal line in

[4] http://www.kyb.tuebingen.mpg.de/bs/people/pgehler/code/index.html

Fig. 4, or lower than the accuracy of LR, it means that negative transfer happens.

(1) *Comparison on the traditional tasks:* From the results in Table 5, we can observe that MLTL obtains the best average performance including the tasks with the accuracy of LR lower and higher than 60%. Additionally, we can find that all the compared transfer algorithms lead to negative transfer. Only MLTL avoids negative transfer on the traditional tasks successfully. The reason that MLTL obtains satisfying performances is two-fold. Firstly, by constructing multiple latent space layers and learning the corresponding distributions, MLTL can utilize more latent factors to reduce the divergence of the distributions among the domains. In particular, when the distribution is dominated by these latent factors and the distribution divergences among domains are so large, MLTL may avoid negative transfer. Secondly, by learning the distributions on the different latent space layers simultaneously, the pluralism of them can be utilized to facilitate more effective transfer learning. In addition, DTL is better than LR and NMTF. This shows that traditional machine learning methods may fail in transfer learning tasks. On the other hand, Tri-TL and HIDC outperform DTL and TCL, the reason may be that learning more distributions can fit different situations on the data distributions.

(2) *Comparison on the new tasks:* For the new tasks, which are more difficult, MLTL obtains the best results once more. As shown in Table 6, although MLTL leads to negative transfer on the new tasks, the number of negative transfer for MLTL is lower than all the compared methods. The reason might be that MLTL constructs the common and specific latent feature spaces as a latent feature space layer. Then more specific latent factors can be utilized to discriminate domains.

(3) *Comparison on the sentiment tasks:* To further validate the adaptability of MLTL, we construct the tasks on sentiment data with two sources and one target domain. In Table 7, we can observe that the performance of MLTL is better than all the compared methods. Particularly, the performance of MLTL is very stable on the tasks with fewer examples. In addition, HIDC is better than DTL, DTL outperforms TCL, Tri-TL and NMTF, and the traditional machine learning method LR outperforms all the compared transfer methods. From the results, we find that all the compared transfer methods which can deal with topic classification tasks fail in the sentiment tasks. Only MLTL is better than LR and exhibits the best performance.

5.4. Effectiveness of Multi-Layer Transfer Learning

In this section, to validate the effectiveness of learning multiple latent space layers, we construct three single layer learning methods, MLTL-L1, MLTL-L2 and MLTL-L3, which are variants of MLTL, and compare them with MLTL. Actually, the algorithms and parameter settings of these single layer methods are very similar to Tri-TL except the number of concepts. From Tables 8–10, we can find that the high-level concept ratios of these methods approximate to 2/2/1. From the results in Table 11, we can find that MLTL obtains the best results on all the tasks. In particular, for the sentiment tasks, all the single latent space layer learning methods cannot compete with the traditional machine learning method Logistic Regression (LR). Only MLTL obtain the best average accuracies on the sentiment tasks. It means that MLTL can deal with not only the topic classification tasks but also the sentiment classification tasks, on which the single layer methods cannot obtain the satisfactory performance.

Moreover, to show the significance of the specific latent spaces, we also construct three algorithms L1-S, L2-S and L3-S, which are variants of MLTL-L1, MLTL-L2 and MLTL-L3 respectively. These algorithms merely construct one common latent space as a latent feature space layer, and the single layer methods MLTL-L1, MLTL-L2 and MLTL-L3 construct both of the common latent space and the specific latent spaces as one latent feature space layer. From the results in Table 12, we can find that MLTL-L1, MLTL-L2 and MLTL-L3 outperform L1-S, L2-S and L3-S respectively. In particular, for the new tasks that the data contains more specific latent factors, MLTL-L1, MLTL-L2 and MLTL-L3 significantly improve the classification accuracies and decrease negative transfer compared with L1-S, L2-S and L3-S respectively. The reason might be that these single layer methods MLTL-L1, MLTL-L2 and MLTL-L3, which construct the specific latent spaces, can utilize more specific latent factors than the single common latent space learning methods to discriminate domains.

5.5. Setting the number of layer

Since it is extremely difficult to formalize the latent factors and quantify the relationships between the latent factors and the latent feature spaces, our method cannot automatically tune the optimal number of the latent feature space layers. Therefore, we evaluate MLTL on our data sets by empirically searching the parameter space for the optimal parameter setting. In this section, we construct two variants of MLTL (MLTL2 and MLTL4), which learns the distributions in two and four latent space layers respectively. From the results in Table 13, we can find that MLTL and MLTL4 outperform MLTL2, and the performance of MLTL is close to MLTL4 on all the tasks. Since the computational complexity is proportional to the number of high-level concepts, we set the number of the latent space layer to three for a more comprehensive effectiveness.

5.6. Parameter sensitivity

In this section, we will investigate the parameter sensitivity of MLTL with nine parameters, $k_{l_1}^{c_1}, k_{l_1}^{c_2}, k_{l_1}^{c_3}, k_{l_2}^{c_1}, k_{l_2}^{c_2}, k_{l_2}^{c_3}, k_{l_3}^{c_1}, k_{l_3}^{c_2}$ and $k_{l_3}^{c_3}$ which represent the numbers of identical, homogeneous and distinct concepts on three latent space layers respectively. The detail description is shown in Table 14. To prove that MLTL is stable when the parameters vary in a widely range, we randomly select 9 combinations of parameter when $k_{l_1}^{c_1} \in [5,15]$, $k_{l_1}^{c_2} \in [5,15]$, $k_{l_1}^{c_3} \in [1,10]$, $k_{l_2}^{c_1} \in [5,15]$, $k_{l_2}^{c_2} \in [5,15]$, $k_{l_2}^{c_3} \in [1,10]$, $k_{l_3}^{c_1} \in [5,15]$, $k_{l_3}^{c_2} \in [5,15]$ and $k_{l_3}^{c_3} \in [1,10]$, then investigate them on 7 randomly selected traditional tasks. From the results shown in Table 14, we can observe that the average accuracy of 9 parameter combinations on each selected task is almost the same as the one using the default parameters, and the variance is very small. Therefore, MLTL is generally not sensitive to the parameters which are selected from the predefined bounds.

5.7. Algorithm convergence

In this section, we check the convergence of MLTL on 6 traditional tasks chose randomly. In Fig. 5, the left and right y-axis indicate the prediction accuracy and the objective value in Eq. (13) respectively, and the x-axis indicates the number of iterations. In Fig. 5, we can observe that the prediction accuracy of MLTL increases with more iterations and the objective value decreases with more iterations conversely, and both of them converge within 200 iterations.

6. Conclusion

Many previous transfer learning algorithms [3–7] showed that the latent high-level concepts, which are related to feature clusters extracted on the raw features, are more appropriate for the text classification across domains than learning from the original features. Additionally, some of these methods [6,7] further proved that the specific latent factors can be used to discriminate domains.

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

However, the methods in [6,7] did not construct the specific latent spaces, thus lots of specific latent factors which exist in the specific latent spaces are ignored. Moreover, these previous approaches [3–7] only construct one common latent space to transfer knowledge, since the set of the latent factors in one latent space is just a subset of all the latent factors, it will ignore some other latent factors to construct the single latent space.

In this paper, we systemically analyze the effectiveness of MLTL, and propose a general cross-domain learning model based on a Non-Negative Matrix Tri-Factorization technology. This model constructs multiple latent space layers that each layer includes both of the common and specific latent feature spaces, and learns the corresponding distributions with the pluralism of them to utilize more latent factors for knowledge transfer. Then an effective algorithm is proposed to derive the solution to the optimization problem. Finally, we conduct comprehensive experiments to show that MLTL outperforms all the compared state-of-art methods. Moreover, the number of layers and the number of feature clusters in each layer in MLTL are tuned manually. In the future, we will investigate automatic approaches to tune these parameters.

Acknowledgment

The authors would like to thank the anonymous reviewers and the editor for their constructive and valuable comments. This work is supported by Grants from the National Natural Science Foundation of China (Nos. 61305063, 61273292, 61303131), the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20130111110011, the Department of Education of Fujian Province (No. JA14129) and the Program for New Century Excellent Talents in Fujian Province University.

References

[1] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[2] J. Blitzer, R. McDonald, F. Pereira, Domain adaptation with structural correspondence learning, In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2006, pp. 120–128.
[3] W.Y. Dai, G.R. Xue, Q. Yang, Y. Yu, Co-clustering based classification for out-of-domain documents, In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 210–219.
[4] F.Z. Zhuang, P. Luo, H. Xiong, Q. He, Y.H. Xiong, Z.Z. Shi, Exploiting associations between word clusters and document classes for cross-domain text categorization, Stat. Anal. Data Min.: ASA Data Sci. J. 4 (1) (2011) 100–114.
[5] M. Long, J. Wang, G. Ding, W. Cheng, X. Zhang, W. Wang, Dual transfer learning, in: Proceedings of the 12th SIAM SDM, 2012, pp. 540–551.
[6] F.Z. Zhuang, P. Luo, C.Y. Du, Q. He, Z.Z. Shi, Triplex transfer learning: exploiting both shared and distinct concepts for text classification, IEEE Trans. Cybern. 44 (7) (2014) 1191–1203.
[7] F.Z. Zhuang, P. Luo, P.F. Yin, Q. He, Z.Z. Shi, Concept learning for cross-domain text classification: a general probabilistic framework, In: Proceedings of the 23th international Joint Conference on Artificial Intelligence (IJCAI), 2013, pp. 1960–1966.
[8] D. Hosmer, S. Lemeshow, Applied Logistic Regression, Wiley, New York, NY, USA, 2000.
[9] T. Hofmann, Unsupervised learning by probabilistic latent semantic analysis, Mach. Learn. 42 (1–2) (2001) 177–196.
[10] T. Li, V. Sindhwani, C. Ding, Y. Zhang, Bridging domains with words: opinion analysis with matrix tri-factorizations, In: Proceedings of the 2010 SIAM International Conference on Data Mining (SDM), 2010, pp. 293–302.
[11] Q. Liu, A.J. Mackey, D.S. Roos, F.C.N. Pereira, Evigan: a hidden variable model for integrating gene evidence for eukaryotic gene prediction, Bioinformatics 24 (5) (2008) 597–605.
[12] Y. Zhu, Y. Chen, Z. Lu, S.J. Pan, G.R. Xue, Y. Yu, Q. Yang, Heterogeneous transfer learning for image classification, In: Proceedings of the 25th AAAI, 2011.
[13] F.Z. Zhuang, P. Luo, Z. Shen, Q. He, Y. Xiong, Z.Z. Shi, H. Xiong, Collaborative dual-plsa: mining distinction and commonality across multiple domains for text classification, In: Proc. of the 19th ACM CIKM, 2010, pp. 359–368.
[14] H. Wang, H. Huang, F. Nie, C. Ding, Cross-language web page classification via dual knowledge transfer using nonnegative matrix tri-factorization, In: Proceedings of the 34th ACM SIGIR, 2011, pp. 933–942.
[15] D.D. Lee, H.S. Seung, Algorithms for non-negative matrix factorization, In: Proceedings of Advances in Neural Information Processing Systems (NIPS), 2000, pp. 556–562.
[16] C. Ding, T. Li, W. Peng, H. Park, Orthogonal nonnegative matrix tri-factorizations for clustering, In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 126–135.
[17] Z. Chen, W.X. Zhang, Domain adaptation with topic correspondence learning, In: Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI), 2013, pp. 1280–1286.
[18] D. Zhang, J. He, Y. Liu, L. Si, R.D. Lawrence, Multi-view transfer learning with a large margin approach, In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 1208–1216.
[19] W.Y. Dai, Q. Yang, G.R. Xue, Y. Yu, Boosting for transfer learning, in: Proceedings of the 24th International Conference on Machine Learning (ICML), 2007, pp. 193–200.
[20] J. Gao, W. Fan, J. Jiang, J.W. Han, Knowledge transfer via multiple model local structure mapping, In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 283–291.
[21] J. Jiang, C.X. Zhai, A two-stage approach to domain adaptation for statistical classifiers. In: Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM), 2007, pp. 401–410.
[22] S. Uguroglu, J. Carbonell, Feature selection for transfer learning, In: Machine Learning and Knowledge Discovery in Databases, 2011, pp. 430–442.
[23] S.J. Pan, J.T. Kwok, Q. Yang, Transfer learning via dimensionality reduction, In: Proceedings of the 23rd AAAI, 2008, pp. 677–682.
[24] M. Long, J. Wang, G. Ding, D. Shen, Q. Yang, Transfer learning with graph co-regularization, In: Proceedings of the 26th AAAI, 2012.
[25] J. Jiang, C.X. Zhai, Instance weighting for domain adaptation in nlp, in: Proceedings of the 45th ACL, 2007, pp. 264–271.
[26] J. Blitzer, M. Dredze, F. Pereira, Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification, In: Proceedings of the 45th ACL, 2007, pp. 440–447.
[27] D. Cook, K.D. Feuz, N.C. Krishnan, Transfer learning for activity recognition: a survey, Knowl. Inf. Syst. 36 (3) (2013) 537–556.
[28] J. Shell, S. Coupland, Fuzzy transfer learning: methodology and application, Inf. Sci. 293 (1) (2014) 59–79.
[29] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[30] V.W. Zheng, S.J. Pan, Q. Yang, J.J. Pan, Transferring multi-device localization models using latent multi-task learning, In: AAAI, Chicago, Illinois, USA, vol. 8, 2008, pp. 1427–1432.
[31] S. Ji, L. Tang, S. Yu, J. Ye, A shared-subspace learning framework for multi-label classification, ACM Trans. Knowl. Discov. Data 4 (2010) 2.
[32] Y. Jiang, F.L. Chung, H. Ishibuchi, Z. Deng, S. Wang, Multitask TSK fuzzy system modeling by mining intertask common hidden structure, IEEE Trans. Cybern. 45 (3) (2015) 548–561.
[33] W. Pan, Q. Yang, Transfer learning in heterogeneous collaborative filtering domains, Artif. Intell. 197 (3) (2013) 39–55.
[34] Z. Deng, Y. Jiang, F.L. Chung, H. Ishibuchi, S. Wang, Knowledge-leverage-based fuzzy system and its modeling, IEEE Trans. Fuzzy Syst. 21 (4) (2013) 597–609.
[35] Z. Deng, Y. Jiang, K.S. Choi, F.L. Chung, S. Wang, Knowledge-leverage-based TSK fuzzy system modeling, IEEE Trans. Neural Netw. Learn. Syst. 24 (8) (2013) 1200–1212.
[36] Z. Deng, K.S. Choi, Y. Jiang, S. Wang, Generalized hidden-mapping ridge regression, knowledge-leveraged inductive transfer learning for neural networks, fuzzy systems and kernel methods, Knowl. Inf. Syst. 44 (12) (2014) 2585–2599.

Jianhan Pan received his M.S. degree in Soft Engineering from University of Electronic Science and Technology of China, in 2012. He is currently an Ph.D. student in Hefei University of Technology. His research interests are data mining and machine learning.

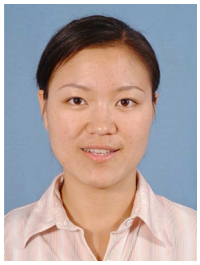
Xuegang Hu received his B.S. degree from the Department of Mathematics at Shangdong University, China, and his M.S. and Ph.D. degrees in Computer Science from Hefei University of Technology, China. Currently, he is a Professor of the School of Computer and Information, Hefei University of Technology, China. He is engaged in research on data mining and knowledge engineering.

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097

Peipei Li received her B.S., M.S. and Ph.D. degrees in Computer Science from Hefei University of Technology, China. Currently, she is a Lecturer of the School of Computer and Information, Hefei University of Technology, China. Her research interests are in the areas of data-stream classification and semi-supervised learning.

Huizong Li received his M.S. degree in Computer Application Technology from Anhui University of Science and Technology, in 2006. He is currently a Ph.D. candidate in Hefei University of Technology, and an Associate Professor in Anhui University of Science and Technology. His research interests are in the areas of knowledge management and social tagging system.

Wei He received his M.S. degree in Computer Application Technology from Hefei University of Technology, in 2010. He is currently an Ph.D. student in Hefei University of Technology. His research interests are data mining and social network.



Yuhong Zhang received her B.S., M.S. and Ph.D. degrees in Computer Science from Hefei University of Technology, China. Currently, she is an Associate Professor of the School of Computer and Information, Hefei University of Technology, China. Her research interests are in the areas of data-stream classification and transfer learning.

Yaojin Lin received the Ph.D. degree in School of Computer and Information from Hefei University of Technology. He currently is an Associate Professor in the School of Computer Science, Minnan Normal University. His research interests include data mining, and granular computing. He has published more than 20 papers in many journals, such as Neurocomputing, Decision Support Systems, Knowledge-Based Systems and Applied Intelligence.

Please cite this article as: J. Pan, et al., Domain adaptation via Multi-Layer Transfer Learning, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.12.097