Transitive Transfer Learning

Ben Tan
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China
btan@cse.ust.hk

Yangqiu Song
Department of Computer Science, University of Illinois at Urbana-Champaign, USA
yqsong@illinois.edu

Erheng Zhong
Personalization Sciences, Yahoo Labs, Sunnyvale, USA
erheng@yahoo-inc.com

Qiang Yang
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China
qyang@cse.ust.hk

ABSTRACT

Transfer learning, which leverages knowledge from source domains to enhance learning ability in a target domain, has been proven effective in various applications. A major limitation of transfer learning is that the source and target domains should be directly related. If there is little overlap between the two domains, performing knowledge transfer between these domains will not be effective. Inspired by human transitive inference and learning ability, whereby two seemingly unrelated concepts can be connected by series of intermediate bridges using auxiliary concepts, in this paper we study a novel learning problem: *Transitive Transfer Learning* (abbreviated to TTL). TTL is aimed at breaking the large domain distances and transferring knowledge even when the source and target domains share few factors directly. For example, when the source and target domains are text and images respectively, TTL can use some annotated images as the intermediate domain to bridge them. To solve the TTL problem, we propose a framework wherein we first select one or more domains to act as a bridge between the source and target domains to enable transfer learning, and then perform the transferring of knowledge via this bridge. Extensive empirical evidence shows that the framework yields state-of-the-art classification accuracies on several classification data sets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Machine Learning

KDD '15, August 11 - 14, 2015, NSW, Australia
 ISBN 978-1-4503-3664-2/15/08 ...$15.00
DOI: http://dx.doi.org/10.1145/2783258.2783295.

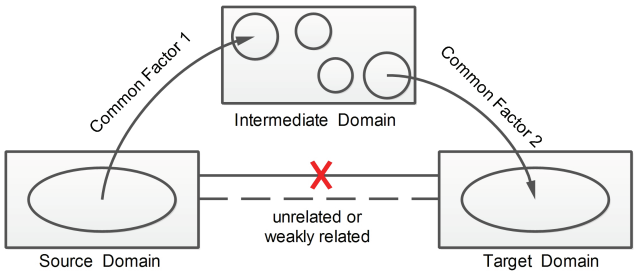


Figure 1: An pictorial illustration of the transitive transfer learning problem. In TTL, the source and target domains have few common factors, but they can be connected by intermediate domains through some underlying factors.

Keywords

Transfer Learning, Transitive Transfer Learning, Nonnegative Matrix Tri-factorizations

1. INTRODUCTION

Transfer learning, which aims to borrow knowledge from source domains to help the learning in a target domain, has been established as one of the most important machine learning paradigms [18]. Various algorithms have been widely used and proven effective in many applications, for example, classification [1, 6], reinforcement learning [24] and recommendation systems [19], and so on. A critical requirement for successful transfer learning is that the source and target domains should be related. This relation can be in the form of related instances, features or models. If no direct relation can be found, forcibly transferring knowledge will not work. In the worst case, it could lead to having no improvement, or even worse performance, in the target domain [21]. This is one of the major limitations of traditional transfer learning. However, as human beings, we naturally have the ability to carry out inference and learning via transitivity [7]. This ability helps humans connect many concepts and transfer knowledge between two seemingly unrelated concepts by introducing a few intermediate concepts as a bridge. For example, after taking a class in elementary computer science,

we may find it easier to transfer the knowledge to theoretical computer science if we have taken an applied algorithm design course in between, since the algorithm course may involve both concepts in programming and theory. Likewise, having learned some basic math, we may find it impossible to directly take a course in convex optimization. However, this becomes feasible when we take an intermediate course in linear algebra and probability. The linear algebra and probability course serves as the intermediate domain for knowledge transfer.

Human ability to conduct transitive inference and learning inspires us to study a novel learning problem known as *Transitive Transfer Learning* (TTL). As illustrated in Figure 1, in TTL, the source and target domains have few common factors, but they can be connected by intermediate domains through some underlining factors. We expect TTL to have wide practical applications. For example, when the source domain is composed of text documents and the target domain contains image data, they share no overlap feature spaces, knowledge learned in text documents can hardly be transferred to images. However, TTL can introduce some annotated images to learn a feature mapping between these two different feature spaces and have a smooth knowledge transfer. In other applications, such as text sentiment classification, all the data have the same feature space, but two group of data may have large distribution gap, TTL can introduce some auxiliary intermediate data to form a transitive knowledge transfer structure with which we can obtain a more versatile sentiment classification system.

In this paper, we propose a learning framework for the TTL problem. The framework is composed of two steps. The first step is to find an appropriate domain to bridge the given source and target domains. The second step is to do effective knowledge transfer among all domains. In the first step, we propose a probability model to select appropriate domains that is able to draw the source and target domains closer, based on domain characteristics such as domain difficulty and pairwise closeness. As data from different domains are collected from different data sources, each pair of domains may have distribution shift. In the second step, considering both of the domain relationship and distribution shift, we propose a transfer learning algorithm that allows to learn overlap features among domains and propagate label information through them. A high-level description of the TTL framework is summarized in Table 1. We give a formal definition of the TTL problem in Section 2, and describe the technical details of these two steps in Sections 3 and 4 respectively.

2. PROBLEM DEFINITION

In the problem, we have labeled source domain data $\boldsymbol{\mathcal{S}}$=$\{(\boldsymbol{x}_i^s, y_i)\}_{i=1}^{n_s}$, unlabeled target domain data $\boldsymbol{\mathcal{T}}$=$\{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$, and k unlabeled intermediate domains $\boldsymbol{D}_j = \{\boldsymbol{x}_i^{d_j}\}_{i=1}^{n_j}$, $j = 1, \ldots, k$, $\boldsymbol{x}^* \in R^{m^*}$ is a m^* dimensional feature vector. The data from different domains could have different dimensions. $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{T}}$ have a large distribution gap, thus directly transferring knowledge between them may cause a substantial performance loss in the target domain. The TTL framework is aimed at finding intermediate domain(s) to bridge $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{T}}$, and minimizing the performance loss in $\boldsymbol{\mathcal{T}}$.

Formally, given a domain distribution gap measure $g(\cdot,\cdot)$, the first step is to find an intermediate domain that satisfies $g(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}}|\boldsymbol{D}_i) < g(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}})$. The second step performs transfer learning from the source domain $\boldsymbol{\mathcal{S}}$ to target domain $\boldsymbol{\mathcal{T}}$ via intermediate domain $\boldsymbol{D}_i$; this is implemented via learning two feature clustering functions $p_{sd}(\boldsymbol{\mathcal{S}}, \boldsymbol{D}_i)$ and $p_{dt}(\boldsymbol{D}_i, \boldsymbol{\mathcal{T}})$, such that the distribution gap of data on common feature clusters selected by $p_{sd}(\boldsymbol{\mathcal{S}}, \boldsymbol{D}_i)$ and $p_{dt}(\boldsymbol{D}_i, \boldsymbol{\mathcal{T}})$ are further reduced. The label information in the source domain is propagated to the intermediate and target data on the selected common feature clusters.

Table 1: The TTL Framework

Input: The source,target and candidate intermediate domains

Step 1: Intermediate domain selection (see Section 3)
Step 2: Transitive knowledge transfer (see Section 4)

Output: Prediction results in the target domain

3. INTERMEDIATE DOMAIN SELECTION

Intermediate domain selection is problem specific, different problems may have different strategies. For example, when the source domain is composed of text data and the target domain is image data, one can crawl some annotated images from Flickr as the intermediate domain data [22]. In other problems when there are multiple candidate intermediate domains, one should propose some selection algorithms according to domain properties. In this paper, we propose an algorithm for text sentiment classification problem as an example. As studied by previous research, domain difficulty [20] and domain distance [3] are two major factors that affect the transfer learning performance between two domains. On one hand, intuitively, if the source domain is less difficult than the intermediate and target domains, the model learned from the source data is highly predictive and is very likely to achieve high performance on the intermediate and target domains as well. On the other hand, if the intermediate domain is able to draw closer the source and target domains than their original distance, then the knowledge transfer process between the source and target domains will have less information loss. Hence, in this paper, we introduce domain complexity [20] and $\mathcal{A}$-distance [3] to estimate domain difficulty and pairwise domain distance respectively. We summarize these measures as follows:

- Domain complexity: domain difficulty measure is problem specific, as different problem may have different feature types. In this paper, we choose domain complexity [14, 20] to measure the difficulty. The domain complexity is calculated as the percentage of long tail features that have low frequency. These long tail features bring in long tail feature distribution and significant feature diversity, thus make automatic machine learning difficult. We calculate the domain complexity as follows:

$$cplx(D) = \frac{|\{x|c(x) < t \times n\}|}{m}, \quad (1)$$

  For non-negative features, $c(x)$ is the number of instances whose feature x is larger than zero. $|\{x|c(x) < t \times n\}|$ is the number of features that appear in less than $t \times n$ instances. In this paper, we measure the domain complexity as the percentage of long tail features that appear in less than 10% instances. For continuous features, we can measure their relative entropy as domain difficulty [20].

- $\mathcal{A}$-distance: The $\mathcal{A}$-distance estimates the distribution difference of two sets of data samples that are drawn from two probability distributions. Practically, given two sets of domain data $\boldsymbol{D}_i$ and $\boldsymbol{D}_j$, we can calculate the $\mathcal{A}$-distance as follows:

$$dis_{\mathcal{A}}(\boldsymbol{D}_i, \boldsymbol{D}_j) = 2(1 - 2 \min_{h \in \mathcal{H}} error(h|\boldsymbol{D}_i, \boldsymbol{D}_j)), \quad (2)$$

Table 2: Domain characteristic features

feature	description
cplx_src (c_1)	source domain complexity
cplx_inter (c_2)	intermediate domain complexity
cplx_tar (c_3)	target domain complexity
$dis^{si}_{\mathcal{A}}$ (c_4)	a_distance between source and intermediate
$dis^{st}_{\mathcal{A}}$(c_5)	a_distance between source and target
$dis^{it}_{\mathcal{A}}$ (c_6)	a_distance between intermediate and target

$\mathcal{H}$ is a hypothesis space, h is the optimal proxy classifier that discriminates data points from different domains. In this paper, we first assign the source data positive labels, and target data negative ones, then use logistic regression as the proxy classifier to estimate the $error(h|\boldsymbol{D}_i, \boldsymbol{D}_j))$ in $\mathcal{A}$-distance. In [3], the authors have been proven that the prediction error of the target domain is bounded by the error of the source domain, the $\mathcal{A}$-distance and some other constant factor.

Given a triple $\boldsymbol{tr} = \{\boldsymbol{\mathcal{S}}, \boldsymbol{D}, \boldsymbol{\mathcal{T}}\}$, we can extract six features as described in Table 2. The first three features summarize individual in-domain characteristics, the last three features capture the pairwise cross domain distances. These features together affect the success probability of a transfer learning algorithm. However, it is impossible to design a universal domain selection criteria, as different problems may have different preferences (weights) on these features. To model the success probability of the introduced intermediate domain, we propose the following logistic function:

$$f(\boldsymbol{tr}) = \delta(\beta_0 + \sum_{i=1}^{6} \beta_i c_i), \tag{3}$$

where $\delta(x) = \frac{1}{1+e^{-x}}$. We estimate the parameters $\boldsymbol{\beta} = \{\beta_0, \cdots, \beta_6\}$ to maximize the log likelihood defined as:

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{i=1}^{t} l^{(i)} \log f(\boldsymbol{tr}_i) + (1 - l^{(i)}) \log(1 - f(\boldsymbol{tr}_i)), \tag{4}$$

$l^{(i)}$ is a binary label, indicating whether the intermediate domain in the ith triple is able to bridge the source and target domains. We get the label by the following strategy. We perform a semi-supervised label propagation algorithm with input $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{T}}$, and obtain a prediction accuracy acc_{st} on the target domain. We also perform the same algorithm with input $\{\boldsymbol{\mathcal{S}}, \boldsymbol{D}, \boldsymbol{\mathcal{T}}\}$, and obtain another accuracy acc_{sit} on the target domain. If $acc_{sit} > acc_{st}$, we set $l^{(i)} = 1$, otherwise, $l^{(i)} = 0$. The label is determined by both the domain characteristics and the propagation model. A sophisticated model may accept more intermediate domains than a simple model. In this paper, we prefer to use a simple model such as KNN that are able to provide us strictly fitted candidates.

We transform the intermediate domain selection problem to a probability estimation problem. A candidate intermediate domain with high $f(\boldsymbol{tr})$ is more likely to be selected.

4. TRANSITIVE KNOWLEDGE TRANSFER

In the first step, an intermediate domain that can bridge the source and target domains has been selected, however, there is still distribution shift among these domains. Thus, in the second step of the TTL framework, we propose a novel transfer learning algorithm that considers both of the transitive relationship and distribution shift among all the domains. The algorithm is based on non-negative matrix tri-factorization that can perform feature clustering and label propagation simultaneously, so we first give some background knowledge.

4.1 Non-negative Matrix Tri-factorization

Non-negative Matrix Tri-factorization (NMTF) is a popular and effective technique for data clustering and classification [10]. In NMTF, the feature-instance matrix is decomposed into three sub-matrices. In general, given a feature-instance matrix $X \in R^{m \times n}$, m is the number of dimensions, n is the number of instances. One can obtain the factorized sub-matrices by solving the optimization problem as follows:

$$\arg min_{F,A,G^T} \quad \mathcal{L} = ||X - FAG^T||, \tag{5}$$

where $||\cdot||$ denotes the Frobenius norm of matrix.

The matrix $F \in R^{m \times p}$ indicates the information of feature clusters and p is the number of hidden feature clusters. The element $F_{i,j}$ indicates the probability that the ith feature belongs to the jth feature cluster.

The matrix $G \in R^{c \times n}$ is the instance cluster assignment matrix and c is the number of instance clusters. If the largest element of the ith row is located in the jth column, it means that the ith instance belongs to the jth instance cluster. In the classification problem, each instance cluster can be regarded as a label class.

$A \in R^{p \times c}$ is the association matrix. c is the number of instance clusters or label classes, for the binary classification problem $c = 2$. The element $A_{i,j}$ is the probability that the ith feature cluster is associated with the jth instance cluster.

4.2 NMTF for Transfer Learning

NMTF is also used as a basic technique for transfer learning algorithms. Given the source and target domains $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{T}}$, X_s and X_t are their feature-instance matrices respectively, one can decompose these two matrices simultaneously, and allow the decomposed matrices share some cross-domain information (sub-matrices). Formally, given two related domains $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{T}}$, their feature-instance matrices can be decomposed simultaneously as follows:

$$\begin{aligned}\mathcal{L}_{ST} &= ||X_s - F_s A_s G_s|| + ||X_t - F_t A_t G_t|| \\ &= \left\| X_s - [F^1, F_s^2] \begin{bmatrix} A^1 \\ A_s^2 \end{bmatrix} G_s^T \right\| + \left\| X_t - [F^1, F_t^2] \begin{bmatrix} A^1 \\ A_t^2 \end{bmatrix} G_t^T \right\|,\end{aligned} \tag{6}$$

where $F^1 \in R_+^{m \times p_1}$ and $A_1 \in R_+^{p_1 \times c}$ contain the common factors shared by the source and target domains. $F_s^2, F_t^2 \in \mathcal{R}_+^{m \times p_2}$ and $A_s^2, A_t^2 \in R_+^{p_2 \times n}$ contain domain-specific information. They are not shared by domains. p_1, p_2 are two parameters that indicate the number of hidden feature clusters. $G_s \in R^{n \times c}$ is the label class matrix and generated from the instance labels $\{y_i | i = 1, \cdots, n\}$ of the source domain $\boldsymbol{\mathcal{S}}$. If the ith instance belongs to jth class, then the (i, j) element in G_s equals to 1, otherwise, it equals to 0. G_s is a constant matrix and keeps unchanged during the factorization process. G_t is the label class matrix of the target domain. Its elements are variables that we want to learn by the matrix decomposition.

From Eq. (6), we can notice that the label information of the source domain is propagated to the target domain through the shared common factors F_1 and A_1.

4.3 The TTL Transfer Learning Algorithm

As shown in Figure 1, the source, intermediate and target domains have a transitive relationship. In other words, the intermediate domain bridges the source and target domains, but has different common factors to them respectively. Hence, to capture these properties, we propose a coupled NMTF algorithm. The proposed

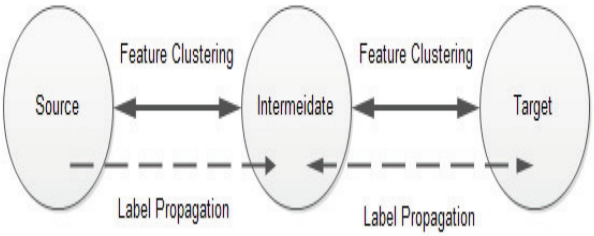


Figure 2: An illustration of the proposed transfer learning algorithm in the TTL framework. The algorithm learns two coupled feature representations by feature clustering, and then propagates the label information from the source to the target domain through the intermediate domain on the coupled feature representation.

transfer learning algorithm is illustrated in Figure 2, and written in Eq. (7)

$$\begin{aligned}\mathcal{L} &= ||X_s - F_s A_s G_s^T|| + ||X_I - F_I A_I G_I^T|| + \\ &\quad ||X_I - F_I^{'} A_I^{'} G_I^T|| + ||X_t - F_t A_t G_t^T|| \\ &= \left\| X_s - [\hat{F}^1, \hat{F}_s^2] \begin{bmatrix} \hat{A}^1 \\ \hat{A}_s^2 \end{bmatrix} G_s^T \right\| + \left\| X_I - [\hat{F}^1, \hat{F}_I^2] \begin{bmatrix} \hat{A}^1 \\ \hat{A}_I^2 \end{bmatrix} G_I^T \right\| + \\ &\quad \left\| X_I - [\tilde{F}^1, \tilde{F}_I^2] \begin{bmatrix} \tilde{A}^1 \\ \tilde{A}_I^2 \end{bmatrix} G_I^T \right\| + \left\| X_t - [\tilde{F}^1, \tilde{F}_t^2] \begin{bmatrix} \tilde{A}^1 \\ \tilde{A}_t^2 \end{bmatrix} G_t^T \right\|. \end{aligned} \tag{7}$$

From the above equation, we can see that the first two terms ($||X_s - F_s A_s G_s^T|| + ||X_I - F_I A_I G_I^T||$) refer to the first feature clustering and label propagation between the source and intermediate domains in Figure 2, the last two terms refer to the second feature clustering and label propagation between the intermediate and target domains. In Eq. (7), it is worth noting that we decompose X_I twice with different decomposition matrices, since X_I shares different knowledge with X_s and X_t respectively. At the same time, we couple these two decomposition processes together by the label matrix G_I. It is reasonable that the instances in the intermediate domain should have the same labels in different decomposition processes. Moreover, if we solve the matrix decomposition by iterative algorithms, in every iteration, each decomposition process is able to consider the feedbacks from the other decomposition. If these two processes are separately solved, the first decomposition process will not consider the results from the second one, and may suffer from the bias problem. In the experiment, we find that the coupled strategy achieves better performance than separated decomposition.

Overall, the proposed learning algorithm fits the transitive relationship among domains. The label information in the source domain is transferred through $\hat{F}_1$ and $\hat{A}^1$ to the intermediate domain, and affects the learning results of G_I. The knowledge on class labels incorporated with G_I from the intermediate domain is further transferred to the target domain through $\tilde{F}_1$ and $\tilde{A}^1$.

As we discussed in Section 4.1, the decomposed matrix F contains the information on hidden feature clusters, indicating the distribution of features on each hidden cluster. Therefore, the summation of each column of F has to be equal to one. The label matrix G indicates the label distribution of each instance. Thus, the summation of each row of G has to be equal to one. Considering these matrix constrains, we obtain the final optimization objective function for the proposed learning algorithm:

$$\begin{aligned} &\arg\min_{F_s,A_s,F_I,A_I,G_I,F_I',A_I',F_t,A_t,G_t} \quad \mathcal{L} \\ &s.t. \\ &\sum_{i=1}^m \hat{F}^1(i,j) = 1, \quad \sum_{i=1}^m \hat{F}_s^2(i,j) = 1, \\ &\sum_{i=1}^m \hat{F}_I^2(i,j) = 1, \quad \sum_{i=1}^m \tilde{F}^1(i,j) = 1, \\ &\sum_{i=1}^m \tilde{F}_I^2(i,j) = 1, \quad \sum_{i=1}^m \tilde{F}_t^2(i,j) = 1, \\ &\sum_{j=1}^c G_I(i,j) = 1 \quad \sum_{j=1}^c G_t(i,j) = 1. \end{aligned} \tag{8}$$

Since the objective function in Eq. (8) is non-convex, it is intractable to obtain the global optimal solution. Therefore, we develop an alternating optimization algorithm to achieve the local optimal solution. We first show the updating rules of matrices $\tilde{F}^1$, $\tilde{F}_I^2$, $\tilde{F}_t^2$, and G_t. We summarize the notations of matrix multiplications in Table 3, and show the updating rules as follows:

$$\begin{aligned} \tilde{F}^1(i,j) &= \tilde{F}^1(i,j) \times \sqrt{\frac{[\tilde{\mathcal{M}}_I^1 + \tilde{\mathcal{M}}_t^1](i,j)}{[\tilde{\mathcal{T}}_I^1 + \tilde{\mathcal{T}}_t^1](i,j)}}, \\ \tilde{F}_I^2(i,j) &= \tilde{F}_I^2(i,j) \times \sqrt{\frac{\tilde{\mathcal{M}}_I^2(i,j)}{\tilde{\mathcal{T}}_I^2(s,t)}}, \\ \tilde{F}_t^2(i,j) &= \tilde{F}_t^2(i,j) \times \sqrt{\frac{\tilde{\mathcal{M}}_t^2(i,j)}{\tilde{\mathcal{T}}_t^2(s,t)}}, \\ G_t(i,j) &= G_t(i,j) \times \sqrt{\frac{[X_t^T F_t A_t](i,j)}{[G_t A_t^T F_t^T F_t A_t](i,j)}}. \end{aligned} \tag{9}$$

From Eq. (8), after the matrices are updated, the constrained matrices have to be normalized as:

$$\begin{aligned} \tilde{F}^1(i,j) &= \frac{\tilde{F}^1(i,j)}{\sum_{i=1}^m \tilde{F}^1(i,j)}, \quad \tilde{F}_I^2(i,j) = \frac{\tilde{F}_I^2(i,j)}{\sum_{i=1}^m \tilde{F}_I^2(i,j)}, \\ \tilde{F}_t^2(i,j) &= \frac{\tilde{F}_t^2(i,j)}{\sum_{i=1}^m \tilde{F}_t^2(i,j)}, \quad G_t(i,j) = \frac{G_t(i,j)}{\sum_{j=1}^c G_t(i,j)}. \end{aligned} \tag{10}$$

The updating rules and normalization methods for other submatrices are similar and are shown in the Appendix. We need not update G_s, which contains the ground-truth label information. We give the procedure of the proposed learning algorithm in Algorithm 1. As shown in Eq. (7) and the Appendix section, the updating rule for G_I is constrained by F_I, F_I', A_I and A_I'. In addition, the sub-matrices $\tilde{F}^1$, $\hat{A}^1$ and, $\tilde{F}^1$, $\tilde{A}^1$ are constrained by X_s, G_s and X_t, G_t respectively. Therefore, the updating rule of G_t is transitively constrained by X_s, G_s and, the discriminative information in the source domain is transitively transferred to the target domain. The updating processes of F_s, F_I, $F_I^{'}$ and F_t refer to the feature clusterings in Figure 2. The updating processes of G_I and G_t refer to the label propagations in Figure 2.

We analyze the convergence property of Eq. (9) with normalization rules in Eq. (10). We first analyze the convergence of $\tilde{F}^1$ with the rest of the parameters fixed. By using the properties of $trace$ operation and frobenius norm $||X||^2 = tr(X^T X) = tr(XX^T)$, we re-formulate the objective function Eq. (8) as a Lagrangian function and keep the terms related to $\tilde{F}^1$:

Table 3: Notations of matrix multiplications

$$\hat{\mathcal{M}}_I^1 = X_I G_I \hat{A}^{1^T} \quad \hat{\mathcal{M}}_I^2 = X_I G_I \hat{A}_I^{2^T} \quad \hat{\mathcal{M}}_t^1 = X_t G_t \hat{A}^{1^T} \quad \hat{\mathcal{M}}_t^2 = X_t G_t \hat{A}_t^{2^T}$$

$$\hat{\mathcal{N}}_I = \hat{F}^1 \hat{A}^1 G_I^T + \hat{F}_I^2 \hat{A}_I^2 G_I^T \qquad \hat{\mathcal{N}}_t = \hat{F}^1 \hat{A}^1 G_t^T + \hat{F}_t^2 \hat{A}_t^2 G_t^T$$

$$\hat{\mathcal{T}}_I^1 = \hat{\mathcal{N}}_I G_I \hat{A}^{1^T} \quad \hat{\mathcal{T}}_t^1 = \hat{\mathcal{N}}_t G_t \hat{A}^{1^T} \quad \hat{\mathcal{T}}_I^2 = \hat{\mathcal{N}}_I G_I \hat{A}_I^{2^T} \quad \hat{\mathcal{T}}_t^2 = \hat{\mathcal{N}}_t G_t \hat{A}_t^{2^T}$$

$$F_t = [\hat{F}_1 \hat{F}_t^2] \qquad A_t = [\hat{A}_1 \hat{A}_t^2]$$

Algorithm 1 The *TTL* Transfer Learning Algorithm

1: **Input**: Source, target, intermediate domains $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{D}}$, the parameters p, and the number of iterations $Iter_{max}$.
2: Initialize the matrices F_s, A_s, F_I, A_I, G_I, F_t, A_t, G_t.
3: **while** $iter < Iter_{max}$ **do**
4: Update the sub-matrices of F_s, A_s, F_I, A_I, F_t, A_t and label matrices G_I, G_t according to the updating rules given in Eq. (9) and Eq. (12) of the Appendix section.
5: Normalize the sub-matrices of F_s, F_I, F_t, and label matrices G_I, G_t according to the normalization rules given in Eq. (10) and Eq. (13) of the Appendix.
6: **end while**
7: **Output**: the predicted results of G_t.

$$\begin{aligned}\mathcal{L}(\tilde{F}^1) = \quad & tr(-2X_I^T \tilde{F}^1 \tilde{A}^1 G_I^T + 2G_I \tilde{A}^{1^T} \tilde{F}^{1^T} \tilde{\mathcal{N}}_I) \\ & + tr(-2X_t^T \tilde{F}^1 \tilde{A}^1 G_t^T + 2G_T \tilde{A}^{1^T} \tilde{F}^{1^T} \tilde{\mathcal{N}}_t) \\ & + tr[\lambda(\tilde{F}^{1^T} \mathbf{1}_m \mathbf{1}_m^T \tilde{F}^1 - 2\mathbf{1}_p \mathbf{1}_m^T \tilde{F}^1)],\end{aligned} \tag{11}$$

where $\lambda \in R^{p \times p}$ is a diagonal matrix. $\mathbf{1}_m$ and $\mathbf{1}_p$ are all-ones vectors with dimension m and p respectively.

LEMMA 1. *Using the update rule in Eq. (9) and normalization rules in Eq. (10), the loss function in Eq. (11) will monotonously decrease.*

The proof of Lemma 1 is shown in the Appendix. The convergence of other terms can be proven in the same way. According to the convergence analysis on the update rules and the multiplicative update rules [13], each update step in Algorithm 1 will not increase Eq. (8). The objective has a lower bounded by zero. The convergence of the proposed transfer learning algorithm is proven.

5. EXPERIMENTS

In this section, we perform three tests. The first test is designed to analyze how the intermediate domain and model parameters affect the performance of the TTL framework, and to evaluate the convergence rate empirically. This is done by conducting experiments on six synthetic text classification tasks generated from the 20Newsgroups data set [1].

The second test is designed to evaluate the TTL framework when the source and target domain data have completely different structures. The experiments are conducted on the text-to-image data set. The intermediate domains for all tasks in the data set are crawled from Flicker.

[1] http://qwone.com/~jason/20Newsgroups/

Finally, the third test is designed to test the efficiency of the intermediate domain selection algorithm and the transfer learning algorithm in the framework. The experiments are conducted on some text sentiment classification tasks [2]. The data from different domains have the same feature space but different distribution. Moreover, there are many candidate intermediate domains for each pair of source and target domains.

5.1 Baseline methods

In the synthetic text classification and sentiment classification tasks, all the data have the same feature space. We compare the proposed framework with three baseline methods to verify the effectiveness.

The first baseline is SVM, which is a classical supervised learning algorithm. We use the linear kernel of SVM with the implementation in LibLinear[3]. The second one is the triplex transfer learning (TriplexTL) algorithm, which is a state-of-the-art transfer learning method implemented with NMTF [32]. The other transfer learning algorithm is LatentMap [25], which is also a state-of-the-art transfer learning algorithm. It draws the joint distribution of two domains closer by mapping the data to a low dimensional latent space. The three baseline methods are tested under two different settings. The first one is direct-transfer. We train the learners based on the labeled data in the source domain and test them directly on the data in the target domain. We use subscript $_{ST}$ to indicate the methods under this setting in the following experiments, for example, TriplexTL$_{ST}$ and LM$_{ST}$. The second setting is a 2-stage transfer learning process. We first apply TriplexTL/LM between the source and the intermediate domain to predict the intermediate domain labels, and then again apply TriplexTL/LM between the intermediate domain and the target domain. The major difference between this naive transitive transfer learning strategy and the proposed transfer learning algorithm is that no iterative feature clustering and label propagation is performed. We use subscript $_{SIT}$ to represent methods under this setting, for instance, TriplexTL$_{SIT}$ and LM$_{SIT}$.

In the text-to-image data set, the data have different feature spaces. The above mentioned baselines cannot handle these data. Hence, we compare TTL with two heterogeneous transfer learning (HTL) algorithms.

The first baseline is co-transfer [17]. It models the problem as a coupled Markov chain with restart. The transition probabilities of the Markov chain is construdture by using the intra-relationship based on affinity metric among data in the source and target domains, and the inter-relationship between the source and target domains based on co-occurrence information of the intermediate do-

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

[3] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

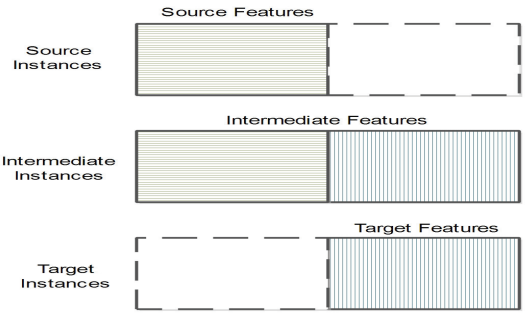
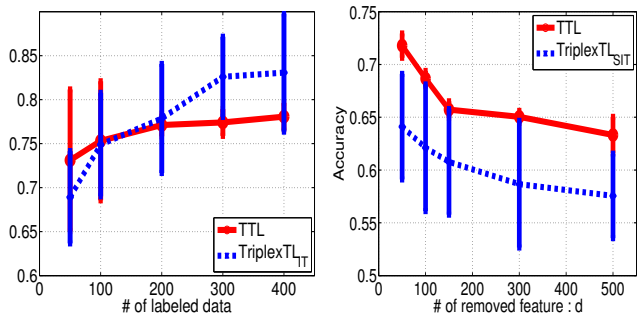


Figure 3: The problem setting of the 20Newsgroup data set.



(a) vary # of labeled data (b) remove d features

Figure 4: Performance with different intermediate domains.

main. The second one is HTLIC [31][4]. It learns a new target feature representation by using data from the source, intermediate and target domain data via the collective matrix factorization technique. A SVM classifier is then learned on the new target feature representation.

All methods in the experiments are performed ten times, and we report their average performances and variances.

5.2 Synthetic text classification tasks

5.2.1 *20Newsgroups Data Set*

The 20Newsgroups is a hierarchical text collection, containing some top categories like 'comp', 'sci', 'rec' and 'talk'. Each category has some sub-categories, such as 'sci.crypt' and 'sci.med'. We use four main categories to generate six tasks, in each of which two top categories are chosen for generating binary categorization. With a hierarchical structure, for each category, all of the subcategories are then organized into three parts, where each part has different subcategories and is of a different distribution. Therefore, they can be treated as the source, intermediate and target domains, respectively. To generate the transitive transfer learning setting, we divide the vocabularies into two separated subsets Set A and Set B. Then, we set the term frequencies of words in Set A of the source domain to zero. Similarly, we set the term frequencies of words in Set B of the target domain to zero. Therefore, the source and target domains have no overlapping words. The problem setting on this data set is illustrated in Figure 3, where the blocks with texture indicate that the features have values. We can see that the source and target domains have no shared features, but they have shared features with the intermediate domain, respectively. Apparently, the intermediate domains here can bridge the generated source and target domains. We give a detailed description of the six tasks in Table 4. The feature dimensions in these tasks range from 2405 to 5984. The number of instances in these tasks are around 7000.

5.2.2 *Performance on synthetic tasks*

In experiments, we compare the proposed framework with the baseline methods on six text classification tasks.

The text classification tasks are very challenging. The source and target domains have no overlapping features. The SVM classifiers trained with labeled source data have almost no discriminative ability on the target data. From the results in Table 5, we can see that the SVM$_{ST}$ classifiers obtain a very bad performance. Likewise, the source classifiers can barely be adapted for the target domain data. Hence, TriplexTL$_{ST}$ and LM$_{ST}$ obtain bad performance also, but better than SVM$_{ST}$. The naive transfer learning algorithms, TriplexTL$_{SIT}$ and LM$_{SIT}$, achieve relative good performance, because they use the intermediate domain data as a bridge to perform a 2-stage knowledge transfer. The proposed TTL framework achieves the best performance. This can be ascribed to the reason that TTL not only bridges the source and target domains by using the intermediate domain data, but also has iterative feature clustering and label propagation loops where the knowledge provided by the source domain can be deeply reshaped and reorganized to be exploited for the target domain.

5.2.3 *Performance with different intermediate domains*

The intermediate domain plays an important role in bridging the source and target domains. Hence, we also conduct some experiments on the "comp-vs-talk" task to test the proposed TTL framework when 1) the amount of labeled intermediate data increases; 2) the connection between the source/target and the intermediate domains becomes weaker.

In the first setting, we compare TTL with TriplexTL$_{IT}$ that transfers knowledge from labeled intermediate domain data to the target data. We vary the amount of labeled intermediate data from 50 to 400. We randomly sample the labeled intermediate domain data ten times, and show the average performance and variance in Figure 4(a). From the results, we can see that the performance of TTL is better than TriplexTL$_{IT}$ when the amount of labeled intermediate domain data is small. However, when there is a large amount of labeled intermediate data, the performance of TriplexTL$_{IT}$ is better. The results are reasonable, because when we have large amount of data that are near and adaptable to the target data, we need not seek help from domains that are far away.

In the second setting, some overlap features in the intermediate domain are removed. We compare the TTL framework with TriplexTL$_{SIT}$. In each comparison experiment, we randomly remove d features ten times, and show the average performance and its variance in Figure 4(b). From the results we can see that the performance decreases as features are removed. The reason is that the connection between the intermediate and source/target domain becomes weaker when more features are removed.

5.2.4 *Model Analysis*

In the Appendix, we have theoretically proven the convergence of the transfer learning algorithm in the TTL framework. Here we test the convergence rate. We conduct an experiment on "comp-vs-talk" task, and set the number of iterations to 100. We show the objective value of Eq. (8) as the dashed line in Figure 5(a), and see that after around five to ten iterations, the objective value experiences almost no change. Similarly, we show the classification accuracy of the target domain of each iteration as the solid line in

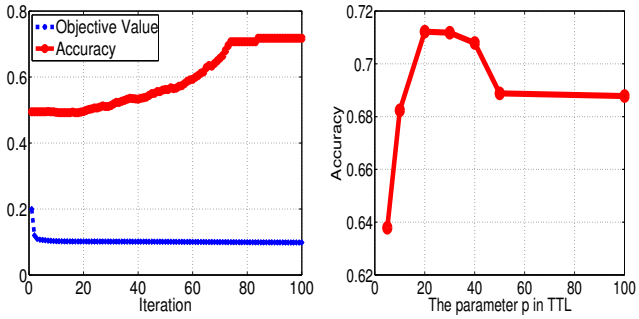
[4] http://www.cse.ust.hk/~yinz/htl4ic.zip

Table 4: Dataset Description

Task	Source	Intermediate	Target
rec-vs-comp	autos : misc	baseball : mac	hockey : windows
rec-vs-talk	autos : guns	motorcycles : mideast	hockey : misc
rec-vs-sci	autos : electronics	motorcycles : med	hockey : space
sci-vs-comp	electronics : graphics	med : misc	space : windows
sci-vs-talk	crypt : guns	electronics : mideast	med : misc
comp-vs-talk	graphics : guns	misc : mideast	windows : politic

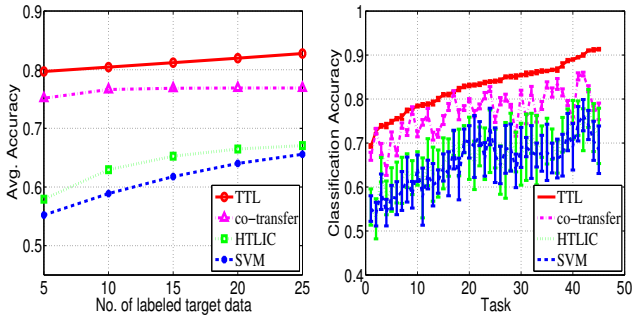
Table 5: Accuracy (%) on the synthetic text classification tasks

	SVM_{ST}	TriplexTL_{ST}	TriplexTL_{SIT}	LM_{ST}	LM_{SIT}	TTL
rec-vs-comp	50.92	53.04 ± 1.87	56.74 ± 4.95	52.23 ± 2.97	55.34 ± 3.75	**57.91** ± 3.27
rec-vs-talk	50.09	59.41 ± 9.74	61.67 ± 7.93	60.11 ± 7.22	60.97 ± 6.53	**68.77** ± 1.61
rec-vs-sci	50.04	51.64 ± 1.44	**51.95** ± 1.70	50.89 ± 2.13	51.23 ± 1.56	**51.95** ± 0.98
sci-vs-comp	50.55	52.14 ± 2.65	55.93 ± 2.39	53.26 ± 2.95	55.29 ± 2.76	**56.26** ± 2.14
sci-vs-talk	50.49	51.57 ± 2.07	52.80 ± 1.66	50.98 ± 2.14	52.69 ± 1.73	**53.15** ± 1.53
comp-vs-talk	50.76	60.90 ± 9.35	64.08 ± 10.19	61.34 ± 9.73	64.58 ± 9.67	**72.22** ± 3.20



(a) Convergence (b) Varying the value of p

Figure 5: Convergence analysis and model parameter analysis.



(a) Average performance (b) Detailed performance

Figure 6: The classification accuracy on the tasks of the text-to-image data set.

Figure 5(a). The results show that there is no change in the performance after 60-80 iterations. The convergence trends on other tasks are similar.

We also analyze the model parameter p. We vary p from 5 to 100 to test how it affects the classification performance. The experiments are also conducted on "comp-vs-talk" task. The results are shown in Figure 5(b), from which we can see that the algorithm achieves better performance when p is between 20 and 40. For different tasks, we can use ten-fold cross validation to choose the value. In this paper, we simply set p to be 30 in the experiments.

5.3 Text-to-image classification tasks

5.3.1 *NUS-WISE data set*

The NUS-WISE data set for heterogeneous transfer learning problem is generated by [17]. It contains 45 text-to-image tasks. Each task is composed of 1200 text documents, 600 images, and 1600 co-occurred text-image pairs. The data in each task are about two different categories, such as "boat" and "flower". Therefore, we can do binary classification for each task. There are 10 categories in the data set, including "bird", "boat", "flower", "food", "rock", "sun", "tower", "toy", "tree" and "car". The text vocabulary size is 500. Each text data is represented by a 500 dimensional bag-of-word vector. For image data, we extract SIFT features [16] and represent each image in a 512 dimensional feature vector. In this data set, our task is to transfer knowledge from source text documents to images through co-occurred text-image pairs.

5.3.2 *Performance on text-to-image tasks*

As HTLIC needs some labeled target domain data to train the SVM classifier, in the text-to-image tasks, we assume all the source domain data and a few target domain data are labeled. We vary the amount of labeled data in the target domain from 5 to 25, and show the average classification accuracies of all the tasks in Fig. 6(a), from which we can see that the performance of each algorithm increases when more labeled target data are used. We can also find that SVM achieves the worst performance, since it considers no auxiliary information. HTLIC and co-transfer achieve better performance than SVM, since they successfully leverage some knowledge from the source domain by using the intermediate domain data. The proposed TTL framework obtains the best performance. The reason is that TTL takes the distribution shift between three domains into account and explicitly exploits the transitively shared knowledge for label propagation from the source to the target domain.

We also report the detailed results on each individual task with 25 labeled target domain data. The classification accuracies and variances on each task are shown in Fig. 6(b). The x-axis indicates

the task and the y-axis represents the classification accuracy. We sort the tasks by the performance of the proposed TTL framework in ascend order. From the results, we can find that TTL is superior to other algorithms on most tasks and is always at the top. In addition, TTL is more stable than other algorithms.

5.4 Sentiment classification tasks

5.4.1 Sentiment Classification Data set

The sentiment classification data set used in our experiment consist of Amazon product reviews on 12 different categories, including "Apparel", "Books", "Camera_&_photo", "DVD", "Electronics", "Health_&_personal_care", "Kitchen_&_housewares", "Music", "Sports_&_outdoors", "Toys_&_games" and "Video". Each product review consists of review text and a sentiment label. The data from different domains have different distributions. For example, reviews in "Kitchen_&_housewares" may have adjectives such as "malfunctioning", "reliable" and "sturdy". However, reviews in the "DVD" domain may have "thrilling", "horrific" and "hilarious". In this data set, the data within each domain are balanced. One half of the data are positive reviews and the other half are negative. The data size in each domain ranges from 2,000 to 20,000. The vocabulary size for each domain is around 20,000. We randomly sample around 2,000 instances for each domain. From the 12 domains, we can generate P_{12}^3=1,320 triples, such as <"Apparel", "Books", "Camera_&_photo"> where "Apparel", "Books" and "Camera_&_photo" are the source, intermediate and target domains respectively. We conduct experiments on all the 1320 triple to evaluate the performance of the proposed intermediate domain selection algorithm. We also conduct experiments on triples that are selected by the intermediate domain selection algorithm to test the proposed transfer learning algorithm in the TTL framework.

5.4.2 Intermediate Domain Selection

In order to evaluate the proposed intermediate domain selection algorithm, we propagate labels from the labeled source domain data to the unlabeled target domain data, and evaluate the prediction accuracy acc_{st} on the target domain data. We also propagate labels from the labeled source domain data to the unlabeled intermediate and target domain data by the same algorithm, and evaluate the prediction accuracy acc_{sit} on the target domain data. In the experiment, we use semi-supervised learning with RBF kernel [30] to do label propagation. If $acc_{sit} > t \times acc_{st}$, $(t > 1.0)$, it means that the intermediate domain data are able to bridge the source and target domain, and we assign a positive label to the triple. Otherwise, we assign a negative label. In the experiment, we set $t = 1.03$, and get 102 positive labels among 1,320 triples.

We then randomly split all the triples into two parts, each part contains the same number of positive and negative triples. The first part is used to train the intermediate domain selection algorithm, the second part is for testing. Since the data are unbalanced, we randomly sampled some negative triples to form a balanced data set. We do the random sampling ten times. Each time, we use 10-fold cross validation to assess the performance of the intermediate domain selection algorithm on the first part. The average accuracy is 0.845 ± 0.034.

5.4.3 Performance on Sentiment Classification Tasks

We also test the proposed transfer learning algorithm in the TTL framework on some triples selected by the intermediate domain selection algorithm with high confidence from the second part. We learn the selection model on the training triples and select 10 triples with highest confidence from the testing triple set. The selected triples are listed in Table 6. Some results are interesting and explainable. For example, "video" domain is able to bridge the "music" and "apparel" domains. Intuitively, most music review words are about sound such as rhythm and melody. Most apparel reviews may talk about the appearance like the color. The video reviews contain both the vocal and visual aspects, and are able to draw the music and apparel domains close.

From the results in Table 6, we can see that Triplex$_{ST}$ has almost the same results as SVM$_{ST}$. The direct transfer learning algorithm here achieves no performance improvement. This is because the source and target domains have large distribution gap. TTL and Triplex$_{SIT}$ are better than Triplex$_{ST}$. We can also see that TTL always achieves the best performance.

6. RELATED WORKS

We discuss two categories of research related to transitive transfer learning: transfer learning and multi-task learning.

Transfer Learning solves the lack of class label problem in the target domain by "borrowing" supervised knowledge from related source domains [18]. There are mainly two typical types of algorithms. The first one is instance based knowledge transfer [9, 23], which selects or adapts the weights of the relevant data from source domains for the target domain. The second one is feature based knowledge transfer [29], that transforms both source and target data into a common feature space where data follow similar distributions. More recently, multi-source transfer learning performs transfer learning with multiple source domains. For instance, the work in [26] extends TrAdaboost [9] by adding a wrapper boosting framework on weighting each source domain. Different from previous transfer learning, transitive transfer learning does not assume that the source domain and the target domain should be related. That means, transitive learning can be more general and more useful when the existing labeled and related source domains are not adequate enough to improve the target domain.

Multi-task Learning algorithms simultaneously learn several tasks together and mutually enhance the classification results of each task [2, 4, 5]. It assumes that different tasks share some natural "compact" representations, such as the information reflected by shared data clusters or subspaces. In practice, for example, classifiers for different tasks can be designed to share some global parameters [11] or even a global classifier [8]. More recently, approaches that learn the relationships between pairwise tasks are also being developed [12, 27, 28]. However, these methods require reasonably large amounts of labeled data for each task to learn the relationship. In contrast, transitive transfer learning works even when both intermediate and target domains are unlabeled. It only assumes that the source domain should have sufficient labeling information to transfer. The intermediate domain serves as a bridge between source and target domains. Even if the intermediate domain is not labeled, the classification information passed from the source domain still contributes to the final classification task through the latent factors learnt in the learning process.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we study a new problem, transitive transfer learning (TTL), which transfers knowledge from a source domain to an indirectly related target domain with the help of some intermediate domains. We propose a TTL framework to solve the problem. The framework first selects one or more intermediate domains to bridge the given source and target domains, and then performs knowledge transfer along this bridge by capturing overlap hidden features among them. The experiments are conducted on three data

Table 6: Accuracy (%) on the Sentiment classification tasks

Source	Intermediate	Target	SVM_{ST}	Triplex_{ST}	Triplex_{SIT}	TTL
music	video	apperal	78.41	78.57 ± 1.84	78.51 ± 1.24	**79.21** ± 1.47
health_&_personal_care	baby	books	74.17	74.15 ± 1.20	74.26 ± 1.21	**75.38** ± 1.51
dvd	toys_&_games	apparel	80.17	80.10 ± 1.46	81.11 ± 1.46	**83.57** ± 1.34
music	toys_&_games	baby	75.94	76.64 ± 1.52	77.64 ± 1.46	**81.22** ± 1.37
books	camera_&_photo	apparel	79.29	80.38 ± 1.34	80.98 ± 1.21	**82.74** ± 1.04
sports_&_outdoors	video	books	71.29	72.25 ± 1.58	73.00 ± 1.67	**76.01** ± 1.05
video	baby	camera_&_photo	77.00	78.43 ± 1.06	79.42 ± 1.03	**81.07** ± 1.06
dvd	kitchen_&_housewares	baby	78.23	78.01 ± 1.13	81.11 ± 1.05	**81.42** ± 1.03
electronics	baby	toys_&_games	81.17	81.60 ± 1.63	81.95 ± 1.49	**82.12** ± 0.09
electronics	baby	kitchen_&_housewares	82.05	83.52 ± 1.02	84.50 ± 1.06	**85.63** ± 1.04

sets, showing that the proposed framework achieves state-of-the-art performance. The convergence of the proposed TTL framework has also been theoretically and experimentally proven.

Future Work As a new learning problem, it raises several issues for further exploration in the future. For example, when the source and target need a string of domains to build a connection, how to find the string of intermediate domains to enable max transfer is a valuable research problem. In addition, extending the algorithm to multiple source domains may be an interesting way to generalize transitive transfer learning to be more powerful.

ACKNOWLEDGMENTS

We thank the support of China National 973 project 2014CB340304 and Hong Kong RGC Projects 621013, 620 812, and 621211. We also thank Yin Zhu, Lili Zhao, Zhongqi Lu, Kaixiang Mo and Ying Wei for discussion.

8. REFERENCES

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, Dec. 2005.

[2] J. Baxter. A bayesian/information theoretic model of learning to learn viamultiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[3] S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, et al. Analysis of representations for domain adaptation. *NIPS*, 19:137, 2007.

[4] S. Ben-David, J. Gehrke, and R. Schuller. A theoretical framework for learning from a pool of disparate data sources. In *KDD*, pages 443–449, 2002.

[5] S. Ben-David and R. Schuller. Exploiting task relatedness for mulitple task learning. In *COLT*, pages 567–580, 2003.

[6] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, Prague, Czech Republic, 2007.

[7] P. E. Bryant and T. Trabasso. Transitive inferences and memory in young children. *Nature*, 1971.

[8] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Multi-task learning for boosting with application to web search ranking. In *KDD*, pages 1189–1198, 2010.

[9] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *ICML*, pages 193–200, 2007.

[10] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *KDD*, pages 126–135. ACM, 2006.

[11] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

[12] Z. Kang, K. Grauman, and F. Sha. Learning with whom to share in multi-task feature learning. In *ICML*, pages 521–528, 2011.

[13] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562. MIT Press, 2000.

[14] C.-K. Lin, Y.-Y. Lee, C.-H. Yu, and H.-H. Chen. Exploring ensemble of models in taxonomy-based cross-domain sentiment classification. CIKM '14, pages 1279–1288, New York, NY, USA, 2014. ACM.

[15] M. Long, J. Wang, G. Ding, W. Cheng, X. Zhang, and W. Wang. Dual transfer learning. In *SDM*, pages 540–551. SIAM, 2012.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[17] M. Ng, Q. Wu, and Y. Ye. Co-transfer learning using coupled markov chains with restart. 2013.

[18] S. J. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, October 2010.

[19] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *IJCAI*, pages 2318–2323, 2011.

[20] N. Ponomareva and M. Thelwall. Biographies or blenders: Which resource is best for cross-domain sentiment analysis? In *CICLing*, pages 488–499. Springer, 2012.

[21] M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Transfer Learning*, volume 898, 2005.

[22] B. Tan, E. Zhong, M. Ng, and Q. Yang. Mixed-transfer: transfer learning over mixed graphs. In *SDM*, 2014.

[23] B. Tan, E. Zhong, W. Xiang, and Q. Yang. Multi-transfer: Transfer learning with multiple views and multiple sources. In *SDM*, 2013.

[24] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *JMLR*, 10:1633–1685, Dec. 2009.

[25] S. Xie, W. Fan, J. Peng, O. Verscheure, and J. Ren. Latent space domain transfer between high dimensional overlapping distributions. In *WWW*, pages 91–100, 2009.

[26] Y. Yao and G. Doretto. Boosting for transfer learning with multiple sources. In *CVPR*, pages 1855–1862, 2010.

[27] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *UAI*, 2010.

[28] Y. Zhang and D.-Y. Yeung. Multi-task boosting by exploiting task relationships. In *ECML/PKDD*, pages 697–710, 2012.
[29] E. Zhong, W. Fan, Q. Yang, O. Verscheure, and J. Ren. Cross validation framework to choose amongst models and datasets for transfer learning. In *ECML/PKDD*, pages 547–562, 2010.
[30] X. Zhu. Semi-supervised learning literature survey. 2005.
[31] Y. Zhu, Y. Chen, Z. Lu, S. J. Pan, G.-R. Xue, Y. Yu, and Q. Yang. Heterogeneous transfer learning for image classification. In *AAAI*, 2011.
[32] F. Zhuang, P. Luo, C. Du, Q. He, and Z. Shi. Triplex transfer learning: exploiting both shared and distinct concepts for text classification. In *WSDM*, pages 425–434, 2013.

Appendix

Table 7: Notations of matrix multiplications

$\hat{\mathcal{M}}_s^1 = X_s G_s \hat{A}^{1^T}$	$\hat{\mathcal{M}}_s^2 = X_s G_s \hat{A}_s^{2^T}$	$\hat{\mathcal{M}}_I^1 = X_I G_I \hat{A}^{1^T}$
$\hat{\mathcal{M}}_I^2 = X_I G_I \hat{A}_I^{2^T}$	$\hat{\mathcal{N}}_s = \hat{F}^1 \hat{A}^1 G_s^T + \hat{F}_s^2 \hat{A}_s^2 G_s^T$	
$\hat{\mathcal{T}}_s^1 = \hat{\mathcal{N}}_s G_s \hat{A}^{1^T}$	$\hat{\mathcal{N}}_I = \hat{F}^1 \hat{A}^1 G_I^T + \hat{F}_I^2 \hat{A}_I^2 G_I^T$	
$\hat{\mathcal{T}}_s^2 = \hat{\mathcal{N}}_s G_s \hat{A}_s^{2^T}$	$\hat{\mathcal{T}}_I^1 = \hat{\mathcal{N}}_T G_T \hat{A}^{1^T}$	$\hat{\mathcal{T}}_I^2 = \hat{\mathcal{N}}_I G_I \hat{A}_I^{2^T}$
$F_I = [\hat{F}_1 \hat{F}_I^2]$	$A_I = [\hat{A}_1 \hat{A}_I^2]$	
$F_I' = [\tilde{F}_1 \tilde{F}_I^2]$	$A_I' = [\tilde{A}_1 \tilde{A}_I^2]$	

We summarize some other matrix multiplication notations in Table 7, and give the update rules for $\hat{F}_s$, $\hat{A}_s$, $\hat{F}_I$ and $\hat{A}_I$ as follow:

$$\hat{F}^1(i,j) = \hat{F}^1(i,j) \times \sqrt{\frac{[\hat{\mathcal{M}}_s^1 + \hat{\mathcal{M}}_I^1](i,j)}{[\hat{\mathcal{T}}_s^1 + \hat{\mathcal{T}}_I^1](i,j)}},$$

$$\hat{F}_s^2(i,j) = \hat{F}_s^2(i,j) \times \sqrt{\frac{\hat{\mathcal{M}}_s^2(i,j)}{\hat{\mathcal{T}}_s^2(i,j)}},$$

$$\hat{F}_I^2(i,j) = \hat{F}_I^2(i,j) \times \sqrt{\frac{\hat{\mathcal{M}}_I^2(i,j)}{\hat{\mathcal{T}}_I^2(i,j)}},$$

$$\hat{A}^1(i,j) = \hat{A}^1(i,j) \times \sqrt{\frac{[\hat{F}^{1^T}(X_s G_s + X_I G_I)](i,j)}{[\hat{F}^{1^T}(\hat{\mathcal{N}}_s G_s + \hat{\mathcal{N}}_I G_I)](i,j)}},$$

$$\hat{A}_s^2(i,j) = \hat{A}_s^2(i,j) \times \sqrt{\frac{[\hat{F}_s^{2^T} X_s G_s](i,j)}{[\hat{F}_s^{2^T} \hat{\mathcal{N}}_s G_s](i,j)}},$$

$$\hat{A}_I^2(i,j) = \hat{A}_I^2(i,j) \times \sqrt{\frac{[\hat{F}_I^{2^T} X_I G_I](i,j)}{[\hat{F}_I^{2^T} \hat{\mathcal{N}}_I G_I](i,j)}},$$

$$G_I(i,j) = G_I(i,j) \times \sqrt{\frac{[X_I^T F_I' A_I' + X_I^T F_I A_I](i,j)}{[G_I A_I'^T F_I'^T F_I' A_I' + G_I A_I^T F_I^T F_I A_I](i,j)}} \tag{12}$$

The normalization methods for $\hat{F}_s$ and $\hat{F}_I$ are:

$$\hat{F}_s(i,j) = \frac{\hat{F}_s(i,j)}{\sum_{i=1}^m \hat{F}_s(i,j)}, \quad \hat{F}_I(i,j) = \frac{\hat{F}_I(i,j)}{\sum_{i=1}^m \hat{F}_I(i,j)}, \tag{13}$$

Convergence Analysis

We first analyze the convergence of $\hat{F}^1$ with the rest parameters are fixed. By using the properties of $trace$ operation and frobenius norm $||X||^2 = tr(X^T X) = tr(X X^T)$, we re-formulate the objective function Eq. (8) as a Lagrangian function and keep the terms related to $\hat{F}^1$:

$$\begin{aligned}\mathcal{L}(\hat{F}^1) = & tr(-2X_s^T \hat{F}^1 \hat{A}^1 G_s^T + 2G_s \hat{A}^{1^T} \hat{F}^{1^T} \hat{\mathcal{N}}_s) \\ & + tr(-2X_I^T \hat{F}^1 \hat{A}^1 G_I^T + 2G_I \hat{A}^{1^T} \hat{F}^{1^T} \hat{\mathcal{N}}_I) \\ & + tr[\lambda(\hat{F}^{1^T} \mathbf{1}_m \mathbf{1}_m^T \hat{F}^1 - 2\mathbf{1}_p \mathbf{1}_m^T \hat{F}^1)],\end{aligned} \tag{14}$$

where $\lambda \in R^{p \times p}$ is a diagonal matrix. $\mathbf{1}_m$ and $\mathbf{1}_p$ are all-ones vectors with dimension m_s and p respectively. The differential of Eq. (14) is:

$$\begin{aligned}\frac{\partial \mathcal{L}(\hat{F}^1)}{\partial \hat{F}^1} = & \quad tr(-2X_s G_s \hat{A}^{1^T} + 2\hat{\mathcal{N}}_s G_s \hat{A}^{1^T}) \\ & + tr(-2X_I Y_t \hat{A}^{1^T} + 2\hat{\mathcal{N}}_t G_t \hat{A}^{1^T}) \\ & + 2\mathbf{1}_m(\mathbf{1}_m^T \hat{F}^1 - \mathbf{1}_p^T)\lambda,\end{aligned} \tag{15}$$

Then, we obtain the temporary updating rule:

$$\hat{F}^1(i,j) = \hat{F}^1(i,j) \times \sqrt{\frac{[X_s G_s \hat{A}^{1^T} + X_I G_I \hat{A}^{1^T} + \mathbf{1}_m \mathbf{1}_p^T \lambda](i,j)}{[\hat{\mathcal{N}}_s G_s \hat{A}^{1^T} + \hat{\mathcal{N}}_I G_I \hat{A}^{1^T} + \mathbf{1}_m \mathbf{1}_m^T \hat{F}^1 \lambda](i,j)}}, \tag{16}$$

As proved in [15], the temporary update rule in Eq. (16) is able to monotonously decrease the Eq. (14). Therefore, there is still one variable λ that needs further calculation. Considering the constrains in Eq. (8), we find that λ is used to satisfy the conditions that the summation of each column of $\hat{F}^1$ has to be equal to one. We use the the normalization method in Eq. (13) to normalize $\hat{F}^1$. The method satisfies the condition regardless of λ. After that, $\mathbf{1}_m \mathbf{1}_p^T \lambda$ is equal to $\mathbf{1}_m \mathbf{1}_m^T \hat{F}^1 \lambda$. By getting rid of the terms that contain λ, we get the final update rule in Eq. (12) that is approximately equal to Eq. (16) in terms of convergence, since both $\mathbf{1}_m \mathbf{1}_p^T \lambda$ and $\mathbf{1}_m \mathbf{1}_m^T \hat{F}^1 \lambda$ are constants. Using update rule in Eq. (12) will also monotonously decrease the value of Eq. (14).

We can use similar methodology to analyze the convergence of the update rules and normalization methods for other terms in Eq. (8).

According to the Multiplicative Update Rules in [13], using the update rules in Eq. (9) and Eq. (12) and using the normalization methods in Eq. (10) and Eq. (13), the value of the objective function in Eq. (8) will not increase. The objective function has a zero lower bound. The convergence of Algorithm 1 is guaranteed.