EigenTransfer: A Unified Framework for Transfer Learning

Wenyuan Dai[†] DWYAK@APEX.SJTU.EDU.CN
Ou Jin[†] KINGOHM@APEX.SJTU.EDU.CN
Gui-Rong Xue[†] GRXUE@APEX.SJTU.EDU.CN
Qiang Yang[‡] QYANG@CSE.UST.HK
Yong Yu[†] YYU@APEX.SJTU.EDU.CN
(†) Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China
(‡) Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

Abstract

This paper proposes a general framework, called EigenTransfer, to tackle a variety of transfer learning problems, e.g. cross-domain learning, self-taught learning, etc. Our basic idea is to construct a graph to represent the target transfer learning task. By learning the spectra of a graph which represents a learning task, we obtain a set of eigenvectors that reflect the intrinsic structure of the task graph. These eigenvectors can be used as the new features which transfer the knowledge from auxiliary data to help classify target data. Given an arbitrary non-transfer learner (e.g. SVM) and a particular transfer learning task, EigenTransfer can produce a *transfer learner* accordingly for the target transfer learning task. We apply EigenTransfer on three different transfer learning tasks, cross-domain learning, cross-category learning and self-taught learning, to demonstrate its unifying ability, and show through experiments that EigenTransfer can greatly outperform several representative non-transfer learners.

1. Introduction

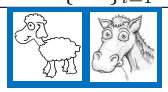
Traditional statistical learning makes a basic assumption that the training data and the test data should be governed by the same underlying distribution. Under this identical-distribution assumption, statistical learning has achieved quite a lot of success. However, the limitation of the identical-distribution assumption is that it isolates the learning tasks in different contexts, even though they may be more or less correlated and can help each other. In this case, transfer learning (Thrun, 1996; Caruana, 1997; Wu & Dietterich, 2004; Raina et al., 2006) can leverage the learned knowledge in one context to enhance the learning in different contexts and thus establish a channel between previous isolated learned tasks.

In the past, a variety of transfer learning tasks have been investigated, including lifelong learning (Thrun, 1996), multi-task learning (Caruana, 1997), cross-domain learning (Wu & Dietterich, 2004; Daumé III & Marcu, 2006), cross-category learning (Raina et al., 2006) and self-taught learning (Raina et al., 2007), to name a few. Most the proposed transfer learning algorithms have achieved great improvement against traditional learning algorithms. However, each of the above specialized works only addresses a particular transfer learning problem, but given the many different approaches and algorithms for transfer learning, there is a lack of general frameworks that can unify the different transfer learning problems and provide corresponding solutions. The focus of this paper is to design such a unifying framework.

In general, transfer learning makes use of the available *auxiliary data* to help the learning on the *target data* that include target training data and target test data. We observe that, to enable transfer learning, the target data and the auxiliary data usually share some common parts and relations between them. For example, in multi-task learning (Caruana, 1997), multiple learning tasks share the same feature space. Thus, we can construct a graph to represent the transfer learning task and model the relations between the target data and the auxiliary data. In this task graph, we use instances, features, and labels to serve as the nodes, while the edges are set based on the relations between the end nodes, connecting the target and auxiliary data in a unified graph structure. By learning the

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

Table 1. The definition of the three representative transfer learning problems: cross-domain learning, cross-category learning, and self-taught learning. The notations and illustrative example have been included in this table. In the example, the pictures in the blue frames are labeled data, while the others are unlabeled.

		Cross-Domain Learning	Cross-Category Learning	Self-taught Learning
Training Data	Instances	$\mathcal{X}_t = \{x_t^{(i)}\}_{i=1}^n$	$\mathcal{X}_t = \{x_t^{(i)}\}_{i=1}^n$	$\mathcal{X}_t = \{x_t^{(i)}\}_{i=1}^n$
	Features	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$
	Labels	$\mathcal{C} = \{c^{(i)}\}_{i=1}^l$	$\mathcal{C}_t = \{c_t^{(i)}\}_{i=1}^l$	$\mathcal{C}_t = \{c_t^{(i)}\}_{i=1}^l$
				
Auxiliary Data	Instances	$\mathcal{X}_a = \{x_t^{(i)}\}_{i=1}^m$	$\mathcal{X}_a = \{x_t^{(i)}\}_{i=1}^m$	$\mathcal{X}_a = \{x_t^{(i)}\}_{i=1}^m$
	Features	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$
	Labels	$\mathcal{C} = \{c^{(i)}\}_{i=1}^l$	$\mathcal{C}_a = \{c_a^{(i)}\}_{i=1}^r$	N/A
				
Test Data	Instances	$\mathcal{X}_u = \{x_u^{(i)}\}_{i=1}^k$	$\mathcal{X}_u = \{x_u^{(i)}\}_{i=1}^k$	$\mathcal{X}_u = \{x_u^{(i)}\}_{i=1}^k$
	Features	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$	$\mathcal{F} = \{f^{(i)}\}_{i=1}^s$
	Labels	N/A, but are expected to be drawn from the same label set as the training data		
				

spectra of this task graph, we can obtain an eigen feature representation for all the nodes in the task graph. This feature representation reflects the intrinsic structure of the task graph that includes the information about target data, auxiliary data, and the relations among them. Under the new feature representation, the knowledge from the auxiliary data can be transferred to help the learning on the target data.

We refer to our framework as EigenTransfer, and conducted extensive experiments to evaluate the effectiveness of our framework on three representative transfer learning problems: cross-domain learning (Daumé III & Marcu, 2006), cross-category learning (Raina et al., 2006), and self-taught learning (Raina et al., 2007). We show that the general framework can indeed model the different transfer learning tasks well. We also show through experimental results that, when compared to non-transfer learners, EigenTransfer can achieve great improvements in general.

2. Problem Formulation

In transfer learning, we have two target data sets: a target *training* data set $\mathcal{X}_t = \{x_t^i\}_{i=1}^n$ with labels, and a target *test* data set $\mathcal{X}_u = \{x_u^i\}_{i=1}^k$ to be predicted. Different from traditional machine learning, we also have an auxiliary data set $\mathcal{X}_a = \{x_t^i\}_{i=1}^m$ to help the target learning.

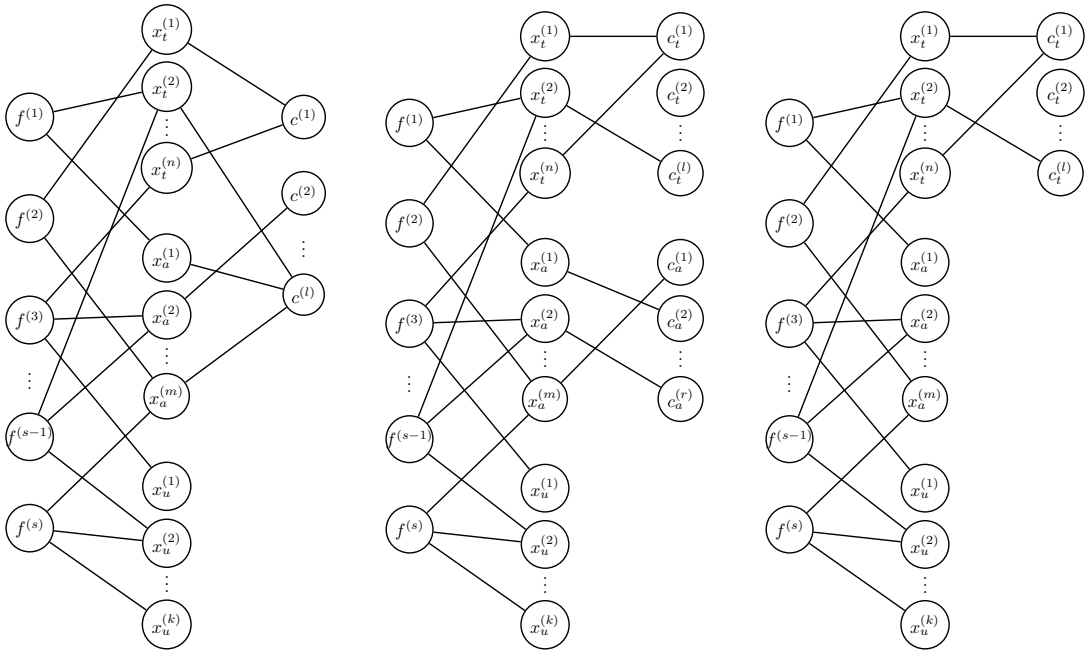
Generally, different transfer learning problems differ in the form of auxiliary data. As shown in Table 1, in cross-domain learning (Daumé III & Marcu, 2006), the auxiliary and the target data (training and test data) share the same categories, i.e. *horse* and *sheep*, but are in different domains, i.e. *real animals* and *cartoon animals*. In cross-category learning (Raina et al., 2006), the auxiliary data are labeled but drawn from different categories from the target data. For example, in Table 1, the target data are about *horse* and *sheep*, while the auxiliary data are about *building* and *sunrise*. In self-taught learning (Raina et al., 2007), the difference from cross-category learning is that the auxiliary data are unlabeled.

A formal definition of these problems including the notations and the illustrative example, is shown in Table 1. In this table, all the instances $x \in \mathcal{X} = \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u$ are represented by the features in the feature space $\mathcal{F} = \{f^{(i)}\}_{i=1}^s$.

The objective of transfer learning is to minimize the prediction error on the test set $\mathcal{X}_u$, with the help of the auxiliary data $\mathcal{X}_a$.

3. Constructing the Task Graphs

In this work, we use spectral learning technique (Chung, 1997) to tackle the transfer learning problems. In spectral learning (Chung, 1997), the input is a weighted graph $G = (V, E)$ to represent the target task, where $V = \{v_i\}_{i=1}^n$ and $E = \{e_{ij}\}_{i,j=1}^n$ represent the node set and the edge set in the graph, respectively. By calculating the eigenvectors of the graph



(a) Cross-Domain Learning (b) Cross-Category Learning (c) Self-taught Learning

Figure 1. The graph construction for three different transfer learning problems: (a) cross-domain learning; (b) cross-category learning; (c) self-taught learning.

G, we obtain the graph spectra for the target task for further learning.

Traditional spectral learning algorithms, such as spectral clustering (Shi & Malik, 2000), usually construct the task graph $G(V, E)$ based on nearest neighbor graph or similarity graph. However, for transfer learning, these two kinds of graphs are no longer capable. Thus, we have to design new graph construction strategies which take into account target data (including training and test data), auxiliary data and the relations among these data.

In this work, we construct a task graph $G(V, E)$ in the following way. As shown in Figure 1, the nodes V in the graph G represent instances, features or class labels, while the edges E represent the relations between these nodes, where the weights of the edges are based on the number of co-occurrences between the end nodes in the target and auxiliary data. Consequently, the task graph $G(V, E)$ contains almost all the information for the transfer learning task, including the target data and the auxiliary data. Usually, the task graph G is sparse, symmetric, real and positive semi-definite. Thus, the spectra of the graph G can be calculated very efficiently.

In the next several subsections, we model three representative transfer learning problems, cross-domain learning (Daumé III & Marcu, 2006), cross-category learning (Raina et al., 2006), and self-taught learning (Raina et al., 2007), with the task graph representation. For many other transfer learning problems, the task graphs can be constructed in similar ways, although they will not be modeled in this paper due to a lack of space.

3.1. Cross-Domain Learning

As shown in Figure 1(a), in the task graph $G(V, E)$ for cross-domain learning, the target instances $x_t^{(i)}$, the auxiliary instances $x_a^{(i)}$, the test instances $x_a^{(i)}$, the features $f^{(i)}$ and the class labels $c^{(i)}$ are used as the nodes. We have

$$V = \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \cup \mathcal{F} \cup \mathcal{C}. \tag{1}$$

The edges E are based on the co-occurrences between any two end nodes in the target training data, the auxiliary data and the target test data. Let e_{ij} denote the weight of the edge between the two nodes v_i and v_j. We have

$$e_{ij} = \begin{cases} \varphi_{v_i,v_j} & v_i \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \wedge v_j \in \mathcal{F} \\ \varphi_{v_j,v_i} & v_i \in \mathcal{F} \wedge v_j \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \\ 1 & v_i \in \mathcal{X}_t \wedge v_j \in \mathcal{C} \wedge C(v_i) = v_j \\ 1 & v_i \in \mathcal{C} \wedge v_j \in \mathcal{X}_t \wedge C(v_j) = v_i \\ 1 & v_i \in \mathcal{X}_a \wedge v_j \in \mathcal{C} \wedge C(v_i) = v_j \\ 1 & v_i \in \mathcal{C} \wedge v_j \in \mathcal{X}_a \wedge C(v_j) = v_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\varphi_{x,f}$ represents the importance of the feature $f \in \mathcal{F}$ appearing in the instance $x \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u$, and

$C(x)$ means the true label of the instance x.

3.2. Cross-Category Learning

Since the problem definition of cross-category learning is different from that of cross-domain learning, its corresponding graph differs consequently. However, the construction of the graph for cross-category learning is still similar as for cross-domain learning. Specifically, we construct a graph $G(V, E)$ to represent the problem of cross-category learning as shown in Figure 1(b), where

$$V = \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \cup \mathcal{F} \cup \mathcal{C}_t \cup \mathcal{C}_a, \tag{3}$$

and

$$e_{ij} = \begin{cases} \varphi_{v_i,v_j} & v_i \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \wedge v_j \in \mathcal{F} \\ \varphi_{v_j,v_i} & v_i \in \mathcal{F} \wedge v_j \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \\ 1 & v_i \in \mathcal{X}_t \wedge v_j \in \mathcal{C}_t \wedge C(v_i) = v_j \\ 1 & v_i \in \mathcal{C}_t \wedge v_j \in \mathcal{X}_t \wedge C(v_j) = v_i \\ 1 & v_i \in \mathcal{X}_a \wedge v_j \in \mathcal{C}_a \wedge C(v_i) = v_j \\ 1 & v_i \in \mathcal{C}_a \wedge v_j \in \mathcal{X}_a \wedge C(v_j) = v_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\varphi_{x,f}$ represents the importance of the feature $f \in \mathcal{F}$ appearing in the instance $x \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u$, and $C(x)$ means the true label of the instance x.

3.3. Self-taught Learning

The only difference between the graph construction of self-taught learning and that of cross-category learning is, in self-taught learning, the nodes with respect to the auxiliary labels $c_a^{(i)}$ are removed, as shown in Figure 1(c), since the auxiliary data are all *unlabeled* in self-taught learning. Let $G(V, E)$ be the task graph for self-taught learning. Then,

$$V = \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \cup \mathcal{F} \cup \mathcal{C}_t. \tag{5}$$

and

$$e_{ij} = \begin{cases} \varphi_{v_i,v_j} & v_i \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \wedge v_j \in \mathcal{F} \\ \varphi_{v_j,v_i} & v_i \in \mathcal{F} \wedge v_j \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u \\ 1 & v_i \in \mathcal{X}_t \wedge v_j \in \mathcal{C}_t \wedge C(v_i) = v_j \\ 1 & v_i \in \mathcal{C}_t \wedge v_j \in \mathcal{X}_t \wedge C(v_j) = v_i \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $\varphi_{x,f}$ represents the importance of the feature $f \in \mathcal{F}$ appearing in the instance $x \in \mathcal{X}_t \cup \mathcal{X}_a \cup \mathcal{X}_u$, and $C(x)$ means the true label of the instance x.

4. Learning Graph Spectra

After the graph construction for transfer learning tasks, we need to learn the spectra of the task graph to form an eigen feature representation. In this work, we use the normalized cut (Shi & Malik, 2000) technique to learn the new feature representation to enable transfer learning.

4.1. Normalized Cut

Given a transfer learning task, based on the graph construction in Section 3, we can obtain a task graph $G(V, E)$ which reflects the information about the transfer learning task, consisting of the target data, the auxiliary data, and the relations between the two kinds of data. Then, we have an adjacency matrix $W \in \mathbb{R}^{n \times n}$ with respect to the graph $G(V, E)$

$$W = \begin{pmatrix} w_{11} & \dots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \dots & w_{nn} \end{pmatrix}, \text{ where } w_{ij} = e_{ij}. \tag{7}$$

Let $D = (d_{ij})_{1 \le i,j \le n}$ be the diagonal matrix with respect to the adjacency matrix W. We have

$$d_{ij} = \begin{cases} \sum_{t=1}^{n} w_{it} & i = j \\ 0 & i \neq j \end{cases}. \tag{8}$$

Then, the graph Laplacian of G can be calculated as follows.

$$L = D - W. \tag{9}$$

The normalized cut (Shi & Malik, 2000) algorithm calculates the first N eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_N$ of the generalized eigenproblem

$$L\mathbf{v} = \lambda D\mathbf{v}. \tag{10}$$

The first N eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_N$ form a new feature representation. Under the new feature representation, traditional learners such as support vector machines (SVM) (Boser et al., 1992) can be used to train classifiers based on the target training data $\mathcal{X}_t$ to predict the labels of target test data $\mathcal{X}_u$. Since the eigen feature representation takes into account the entire transfer learning task, traditional learners under the eigen feature representation are *wrapped* into transfer learners.

The detailed description for our algorithm EigenTransfer is presented in Algorithm 1. In our algorithm, the non-zero occurrences in L is linear to the input, and thus L is usually very sparse. Moreover, L is real, symmetric and positive semi-definite. To solve the eigenproblem for this kind of matrices, we used the *Lanczos algorithm* (Saad, 1992) in this work. Generally, the computational cost of Lanczos algorithm is linear to the input, and thus our algorithm EigenTransfer is efficient and scalable.

Algorithm 1 EigenCluster: a unified framework for transfer learning

Input: a target clustering task, including the target data set $\mathcal{X}_t = \{x_t^{(i)}\}_{i=1}^n$, the auxiliary data set $\mathcal{X}_a = \{x_a^{(i)}\}_{i=1}^m$, and the test data set $\mathcal{X}_u = \{x_u^{(i)}\}_{i=1}^k$.
Output: classification result on $\mathcal{X}_u$.

1: Construct the task graph $G(V, E)$ based on the target clustering task (c.f. Section 3).
2: Calculate the adjacent matrix W based on the task graph $G(V, E)$.
3: $D \leftarrow \text{diag}(We)$; $L \leftarrow D - W$.
4: Calculate the first N eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_N$ which satisfy the generalized eigenproblem: $L\mathbf{v} = \lambda D\mathbf{v}$.
5: Let U be the matrix with $\mathbf{v}_1, \ldots, \mathbf{v}_N$ as columns.
6: **for** each $x_t^{(i)} \in \mathcal{X}_t$ **do**
7: Let $y_t^{(i)}$ be the corresponding row in U with respect to $x_t^{(i)}$.
8: **end for**
9: Train a classifier based on $\mathcal{Y}_t = \{y_t^{(i)}\}_{i=1}^n$ instead of $\mathcal{X}_t = \{x_t^{(i)}\}_{i=1}^n$ using a traditional classification algorithm, e.g. SVM (Boser et al., 1992), and then classify the test data $\mathcal{X}_u = \{x_u^{(i)}\}_{i=1}^k$.

5. Experiments

In this section, we empirically evaluate our algorithm on three different transfer learning problems: cross-domain learning (Daumé III & Marcu, 2006), cross-category learning (Raina et al., 2006), and self-taught learning (Raina et al., 2007). We show that, our framework EigenTransfer can greatly outperform non-transfer learners in general.

5.1. Cross-Domain Learning

The first experiment was conducted for cross-domain learning. We used three data sets, SRAA[1], 20 Newsgroups (Lang, 1995) and Reuters-21578[2], to generate the cross-domain learning tasks.

All three data sets have hierarchical structures. We used the top directories as categories, while splitting the data into target and auxiliary data sets based on sub-directories which we refer to as *domains*. As a result, the target and auxiliary data share the same categories, i.e. top directories, but they belong to different domains, i.e. sub-directories. The description of the data sets for cross-domain learning is presented in Table 2.

[1]http://www.cs.umass.edu/~mccallum/code-data.html
[2]http://www.daviddlewis.com/resources/testcollections/

Table 2. The description of the data sets used for evaluating cross-domain learning. Due to the space limitation, we omit some details of sub-directories.

Data Set	Target Data	Auxiliary Data
cdl-sraa1	real-aviation simulated-aviation	real-auto simulated-auto
cdl-sraa2	simulated-auto simulated-aviation	real-auto real-aviation
cdl-20ng1	rec.*, talk.*	rec.*, talk.*
cdl-20ng2	rec.*, sci.*	rec.*, sci.*
cdl-20ng3	comp.*, talk.*	comp.*, talk.*
cdl-20ng4	comp.*, sci.*	comp.*, sci.*
cdl-20ng5	comp.*, rec.*	comp.*, rec.*
cdl-20ng6	sci.*, talk.*	sci.*, talk.*
cdl-reuters1	orgs.*, places.*	orgs.*, places.*
cdl-reuters2	people.*, places.*	people.*, places.*
cdl-reuters3	orgs.*, people.*	orgs.*, people.*

We evaluated three baseline classifiers, naive Bayes classifier (NBC) (Lewis, 1992), support vector machine (SVM) (Boser et al., 1992), and transductive support vector machine (TSVM) (Joachims, 1999). For each baseline method, we use EigenTransfer to form an eigen feature representation, and then apply the baseline learner to the target data under the eigen feature representation. Therefore, as shown in Table 3, for each baseline method, we have two versions, "Non-Transfer" and "EigenTransfer". Here, "Non-Transfer" means to simply apply the baseline method to the original data; "EigenTransfer" means to apply the baseline method to the data in the new feature representation learned by EigenTransfer.

The performance in Table 3 is measured in error rate by averaging 10 random repeats on each data set by each evaluation method. For each repeat, we randomly selected 30 instances per category as the target training data, in order to simulate the setting for transfer learning where the target labeled data are few. We also include the standard deviations of the repeats in this table. The *negative transfer* cases, for when transfer learning lowers the learning performance, have been set to *Italic* font. From this table, we can see that in most cases, our framework EigenTransfer can perform better than non-transfer learners by a large margin. However, EigenTransfer cannot prevent negative transfer completely, a reasonable fact that is also pointed out in several previous transfer learning literatures such as (Caruana, 1997; Rosenstein et al., 2005).

5.2. Cross-Category Learning

In the experiments for cross-category learning, we used two data sets, the 20 Newsgroups (Lang, 1995) data set and the *ohscal* data set from OHSUMED (Hersh et al., 1994), to generate the tasks.

Table 3. The experimental results in error rate for cross-domain learning. The results are the averages of 10 random repeats, as well as their standard deviations. The results with *Italic* font means negative transfer. All the methods were well-tuned using 10-fold cross-validation.

Data Set	NBC		SVM		TSVM	
	Non-Transfer	EigenTransfer	Non-Transfer	EigenTransfer	Non-Transfer	EigenTransfer
cdl-sraa1	0.271 ± 0.032	0.146 ± 0.021	0.205 ± 0.034	0.097 ± 0.023	0.164 ± 0.011	0.063 ± 0.016
cdl-sraa2	0.251 ± 0.045	0.170 ± 0.028	0.213 ± 0.064	0.059 ± 0.009	0.129 ± 0.014	0.047 ± 0.001
cdl-20ng1	0.273 ± 0.068	0.060 ± 0.008	0.209 ± 0.034	0.028 ± 0.005	0.201 ± 0.060	0.097 ± 0.004
cdl-20ng2	0.150 ± 0.013	0.061 ± 0.020	0.145 ± 0.050	0.026 ± 0.016	0.071 ± 0.007	0.061 ± 0.002
cdl-20ng3	0.199 ± 0.019	0.079 ± 0.034	0.194 ± 0.059	0.030 ± 0.008	0.129 ± 0.129	0.025 ± 0.014
cdl-20ng4	0.266 ± 0.025	0.082 ± 0.032	0.204 ± 0.038	0.074 ± 0.048	0.113 ± 0.052	0.092 ± 0.075
cdl-20ng5	0.180 ± 0.020	0.065 ± 0.027	0.119 ± 0.038	0.029 ± 0.005	0.041 ± 0.002	0.022 ± 0.001
cdl-20ng6	0.160 ± 0.033	0.050 ± 0.035	0.086 ± 0.020	0.013 ± 0.002	0.043 ± 0.002	0.033 ± 0.004
cdl-reuters1	0.357 ± 0.049	0.232 ± 0.054	0.240 ± 0.020	*0.252±0.037*	0.201 ± 0.033	*0.248±0.040*
cdl-reuters2	0.342 ± 0.065	0.277 ± 0.058	0.218 ± 0.027	0.217 ± 0.028	0.219 ± 0.045	0.202 ± 0.027
cdl-reuters3	0.299 ± 0.024	0.249 ± 0.024	0.259 ± 0.044	0.219 ± 0.017	0.231 ± 0.067	0.215 ± 0.024
average	0.250 ± 0.036	0.134 ± 0.031	0.190 ± 0.039	0.095 ± 0.018	0.140 ± 0.038	0.101 ± 0.019

Table 4. The description of the data sets used for evaluating cross-category learning.

Data Set	Target Data
ccl-20ng1	comp.graphics, rec.sport.baseball
ccl-20ng2	comp.sys.ibm.pc.hardware, rec.sport.baseball
ccl-20ng3	sci.crypt, talk.politics.guns
ccl-20ng4	alt.atheism, sci.space
ccl-20ng5	sci.med, talk.politics.mideast
ccl-ohs1	Carcinoma, Pregnancy
ccl-ohs2	Prognosis, Receptors
ccl-ohs3	In-Vitro, Molecular-Sequence-Data
ccl-ohs4	Antibodies, DNA
ccl-ohs5	DNA, In-Vitro

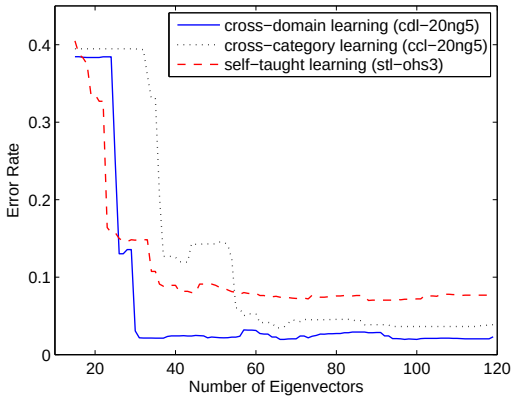


Figure 2. The influence of the number of eigenvectors used to form the eigen feature representation.

The 20 Newsgroups data set has 20 categories, and the oshcal data set contain 10 categories. We used the following strategy to generate cross-category learning tasks. For each task, we selected some categories as the target data, including target training data and test data. The other categories in the data set were used as auxiliary labeled data. Table 4 shows the description of the cross-category learning tasks in our experiments. In this table, we only listed the target categories. For each task, the categories that have not been listed in this table were used as the auxiliary ones.

Table 5 shows the experimental results for the cross-category learning when there are 30 labeled data per category in the target training set. The format of this table is the same as that of cross-domain learning (c.f. Table 3 and Section 5.1). From the table, we can see that, in general, EigenTransfer can improve the non-transfer learners when solving cross-category learning problems, resulting in few negative transfer cases.

5.3. Self-taught Learning

The experiments for self-taught learning were also conducted on the 20 Newsgroups (Lang, 1995) data set, and the ohscal data set from OHSUMED (Hersh et al., 1994). The only difference from the data sets of cross-category learning in Section 5.2 is that we removed all the labels from the auxiliary data. Thus, for the details of the data sets used for self-taught learning, we can refer to Table 4.

Table 6 shows the experimental results with respect to self-taught learning when there are 30 labeled data per category in the target training set. From this table, we can see that although there are two negative transfer cases, generally, EigenTransfer can greatly outperform non-transfer learners in the situation of self-taught learning as well.

5.4. Eigenvectors

In EigenTransfer, there is only one parameter to set which is the number of eigenvectors used to form the eigen-feature representation. In the experiments in Sections 5.1, 5.2 and 5.3, we tuned this number using a 10-fold cross validation. However, we also tested

Table 5. The experimental results in error rate for cross-category learning. The results are the averages of 10 random repeats, as well as their standard deviations. The results with *Italic* font means negative transfer. All the methods were well-tuned using 10-fold cross-validation.

Data Set	NBC		SVM		TSVM	
	Non-Transfer	EigenTransfer	Non-Transfer	EigenTransfer	Non-Transfer	EigenTransfer
ccl-20ng1	0.153 ± 0.041	0.037 ± 0.018	0.120 ± 0.030	0.021 ± 0.012	0.045 ± 0.006	0.016 ± 0.009
ccl-20ng2	0.130 ± 0.054	0.041 ± 0.019	0.125 ± 0.043	0.020 ± 0.006	0.032 ± 0.008	0.012 ± 0.004
ccl-20ng3	0.181 ± 0.052	0.064 ± 0.022	0.077 ± 0.024	0.017 ± 0.011	0.037 ± 0.005	0.029 ± 0.015
ccl-20ng4	0.188 ± 0.039	0.056 ± 0.012	0.090 ± 0.019	0.031 ± 0.011	0.060 ± 0.003	0.048 ± 0.015
ccl-20ng5	0.256 ± 0.052	0.132 ± 0.039	0.174 ± 0.031	0.084 ± 0.022	0.121 ± 0.023	0.080 ± 0.013
ccl-ohs1	0.204 ± 0.053	0.057 ± 0.012	0.083 ± 0.019	0.042 ± 0.010	0.202 ± 0.001	0.199 ± 0.003
ccl-ohs2	0.088 ± 0.013	0.052 ± 0.015	0.094 ± 0.040	0.040 ± 0.008	0.066 ± 0.002	0.058 ± 0.001
ccl-ohs3	0.156 ± 0.020	0.125 ± 0.032	0.130 ± 0.036	0.080 ± 0.025	0.081 ± 0.005	*0.082±0.013*
ccl-ohs4	0.286 ± 0.025	0.240 ± 0.033	0.235 ± 0.047	0.176 ± 0.029	0.217 ± 0.039	0.201 ± 0.036
ccl-ohs5	0.217 ± 0.026	0.184 ± 0.046	0.186 ± 0.030	0.144 ± 0.026	0.179 ± 0.008	0.187 ± 0.017
average	0.186 ± 0.038	0.099 ± 0.025	0.131 ± 0.032	0.065 ± 0.016	0.104 ± 0.010	0.091 ± 0.013

Table 6. The experimental results in error rate for self-taught learning. The results are the averages of 10 random repeats, as well as their standard deviations. The results with *Italic* font means negative transfer cases. All the methods were well-tuned using 10-fold cross-validation.

Data Set	NBC		SVM		TSVM	
	Non-Transfer	EigenTransfer	Non-Transfer	EigenTransfer	Non-Transfer	EigenTransfer
stl-20ng1	0.138 ± 0.050	0.039 ± 0.012	0.112 ± 0.033	0.025 ± 0.018	0.041 ± 0.005	0.014 ± 0.005
stl-20ng2	0.105 ± 0.016	0.048 ± 0.021	0.104 ± 0.037	0.018 ± 0.008	0.032 ± 0.006	0.015 ± 0.004
stl-20ng3	0.193 ± 0.036	0.116 ± 0.048	0.079 ± 0.021	0.041 ± 0.010	0.039 ± 0.004	*0.070±0.050*
stl-20ng4	0.223 ± 0.048	0.052 ± 0.017	0.111 ± 0.034	0.026 ± 0.014	0.061 ± 0.004	0.045 ± 0.006
stl-20ng5	0.292 ± 0.042	0.126 ± 0.027	0.178 ± 0.031	0.092 ± 0.017	0.133 ± 0.033	0.113 ± 0.105
stl-ohs1	0.180 ± 0.049	0.083 ± 0.046	0.066 ± 0.022	0.061 ± 0.026	0.203 ± 0.001	0.200 ± 0.005
stl-ohs2	0.084 ± 0.007	0.063 ± 0.033	0.082 ± 0.032	0.035 ± 0.004	0.065 ± 0.002	0.056 ± 0.005
stl-ohs3	0.153 ± 0.015	0.123 ± 0.029	0.132 ± 0.018	0.082 ± 0.019	0.080 ± 0.004	*0.085±0.019*
stl-ohs4	0.298 ± 0.050	0.248 ± 0.055	0.209 ± 0.038	0.189 ± 0.041	0.230 ± 0.027	0.208 ± 0.023
stl-ohs5	0.220 ± 0.033	0.170 ± 0.032	0.189 ± 0.031	0.133 ± 0.018	0.180 ± 0.021	0.179 ± 0.020
average	0.189 ± 0.035	0.107 ± 0.032	0.126 ± 0.030	0.070 ± 0.017	0.106 ± 0.011	0.098 ± 0.024

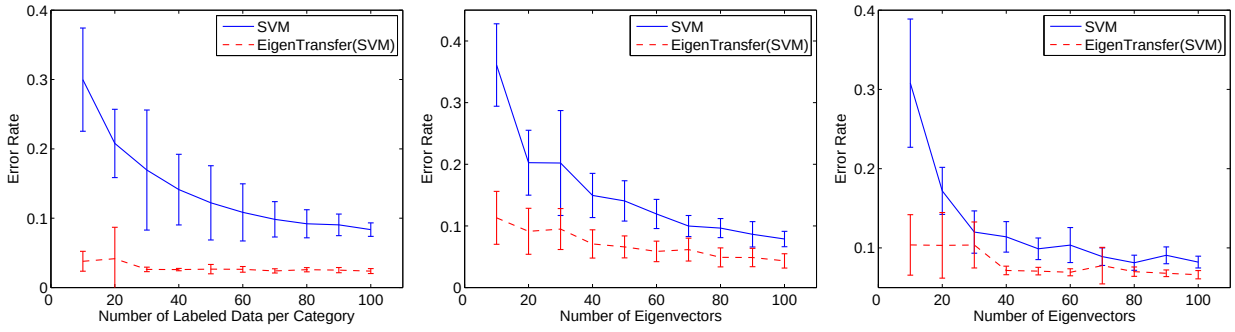
the sensitivity of this parameter. In Figure 2, we varied the number of eigenvectors and tested the performance of EigenTransfer on three data sets, *cdl-20ng5*, *ccl-20ng5* and *stl-ohs3*. From the figure, we can see that when the number of eigenvectors is greater than 60, the performance of EigenTransfer is very stable. In our experiments, based on cross validation, the number of eigenvectors is mostly set to the values between 40 to 120.

5.5. Labeled Target Data

In Sections 5.1, 5.2 and 5.3, we have shown the results when there are 30 labeled data per target category. However, we also want to see the influence of different sizes of target training data. In Figure 3, we varied the number of labeled target training data instances per category from 10 to 100, and show the performance of EigenTransfer on three data sets *cdl-20ng5*, *ccl-20ng5* and *stl-ohs3*, respectively. From the figure, we can see EigenTransfer always outperforms the non-transfer learner SVM. The less the target training data are, the more improvement EigenTransfer can achieve. Moreover, EigenTransfer achieves the most significantly improvements on cross-domain learning, and obtains the least significantly improvements on self-taught learning. We believe this is because the quality of auxiliary data in cross-domain learning is better than that in cross-category learning, which is better than that in self-taught learning.

6. Conclusions and Discussions

In this paper, we proposed a general transfer learning framework, which can model a variety of existing transfer learning problems and solutions. In our framework, first, a task graph is constructed to represent the transfer learning task. Then, we learn an eigen feature representation from the task graph based on spectral learning theory (Chung, 1997). Under the new feature representation, the knowledge from the auxiliary data tends to be transferred to help the target learning. Our experimental results show that the EigenTransfer framework can unify a number of well known transfer learning problems, can be used to wrap any



(a) (b) (c)

Figure 3. The influence of different number of target training data: (a) cross-domain learning on the *cdl-20ng5* data set; (b) cross-category learning on the *ccl-20ng5* data set; (c) self-taught learning on the *stl-ohs3* data set.

non-transfer learning algorithms for a transfer learning task that we consider, and can greatly outperform non-transfer learners in many experiments.

In our work, the task graph is constructed to fully reflect the transfer learning task, including target data, auxiliary data, and their commonness. There is little information loss during the graph construction. The graph spectra can represent the intrinsic structure of the task graph and the transfer learning task. This is the rationale of EigenTransfer being an effective transfer learning approach.

Acknowledgement Wenyuan Dai and Gui-Rong Xue are supported by the grants from National Natural Science Foundation of China (NO. 60873211). We also thank the anonymous reviewers for their greatly helpful comments.

References

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual Workshop on Computational Learning Theory* (pp. 144–152).

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

Chung, F. R. K. (1997). *Spectral Graph Theory*. American Mathematical Society.

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

Hersh, W., Buckley, C., Leone, T. J., & Hickam, D. (1994). Ohsumed: an interactive retrieval evaluation and new large test collection for research. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 192–201).

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Proceedings of the 16th International Conference on Machine Learning* (pp. 200–209).

Lang, K. (1995). Newsweeder: Learning to filter netnews. *Proceedings of the 12th International Conference on Machine Learning* (pp. 331–339).

Lewis, D. D. (1992). *Representation and learning in information retrieval*. Doctoral dissertation, Amherst, MA, USA.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. *Proceedings of the 24th International Conference on Machine Learning* (pp. 759–766).

Raina, R., Ng, A. Y., & Koller, D. (2006). Constructing informative priors using transfer learning. *Proceedings of the 23rd International Conference on Machine Learning* (pp. 713–720).

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (2005). To transfer or not to transfer. *NIPS 2005 Workshop on Transfer Learning*.

Saad, Y. (1992). *Numerical methods for large eigenvalue problems*. Oxford rd, Manchester, UK.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 888–905.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? *Advances in Neural Information Processing Systems 8* (pp. 640–646).

Wu, P., & Dietterich, T. G. (2004). Improving svm accuracy by training on auxiliary data sources. *Proceedings of the 21st International Conference on Machine Learning* (pp. 110–117).