Graph-based Transfer Learning

Jingrui He
School of Computer Science
Carnegie Mellon University
jingruih@cs.cmu.edu

Yan Liu
Predictive Modeling Group
IBM Research
liuya@us.ibm.com

Richard Lawrence
Predictive Modeling Group
IBM Research
ricklawr@us.ibm.com

ABSTRACT

Transfer learning is the task of leveraging the information from labeled examples in some domains to predict the labels for examples in another domain. It finds abundant practical applications, such as sentiment prediction, image classification and network intrusion detection. In this paper, we propose a graph-based transfer learning framework. It propagates the label information from the source domain to the target domain via the example-feature-example tripartite graph, and puts more emphasis on the labeled examples from the target domain via the example-example bipartite graph. Our framework is semi-supervised and nonparametric in nature and thus more flexible. We also develop an iterative algorithm so that our framework is scalable to large-scale applications. It enjoys the theoretical property of convergence. Compared with existing transfer learning methods, the proposed framework propagates the label information to both the features irrelevant to the source domain and the unlabeled examples in the target domain via the common features in a principled way. Experimental results on 3 real data sets demonstrate the effectiveness of our algorithm.

Categories and Subject Descriptors

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

General Terms

Algorithm, experimentation

Keywords

Transfer learning, graph-based

1. INTRODUCTION

Transfer learning refers to the process of leveraging the information from a source domain to train a better classifier for a target domain. Typically there are plenty of labeled examples in the source domain, whereas very few or no labeled examples in the target domain. Transfer learning is of key importance in many real applications. For example, in sentiment analysis, we may have many labeled movie reviews (labels obtained according to the movie ratings), but we are interested in analyzing the polarity of reviews about an electronic product [4]; in face recognition, we have many training images under certain lightening and occlusion conditions based on which a model is trained, but practically the model will be used under totally different conditions [14].

Generally speaking, transfer learning can follow one of the following three scenarios:

1. The source domain and the target domain have the same feature space and the same feature distribution, and only the labeling functions are different, such as multi-label text classification [24];
2. The source domain and the target domain have the same feature space, but the feature distribution and the labeling functions are different, such as sentiment classification for different purposes [4];
3. The source domain and the target domain have different feature space, feature distribution and labeling functions, such as verb argument classification [10].

In this paper, we focus on the second scenario, which sometimes is formalized as the problem that the training set and the test set have different feature distribution [7].

The main contribution of this paper is to develop a graph-based transfer learning framework based on separate constructions of a tripartite graph (labeled examples - features - unlabeled examples) and a bipartite graph (labeled examples - unlabeled examples). By propagating the label information from labeled examples (mostly from the source domain) to unlabeled examples (from the target domain) via the features on the tripartite graph, and by imposing domain related constraints on the bipartite graph, we are able to learn a classification function that takes values on all the unlabeled examples in the target domain. Finally, these examples are labeled according to the sign of the function values. The proposed framework is semi-supervised since it makes use of unlabeled examples to help propagate the label information. Furthermore, in the second transfer learning scenario (which we are interested in), the labeling functions in different domains may be closely related to the feature distribution; thus unlabeled examples are helpful in constructing the classifiers. However, our framework is different from traditional semi-supervised learning due to the fact that labeled examples from different domains are treated differently in order to construct an accurate classifier in the target domain, whereas in traditional semi-supervised learning, all the labeled examples are treated in the same way. The framework is also non-parametric in nature, which makes it more flexible compared with parametric models.

The proposed transfer learning framework is fundamentally different from existing graph-based methods. For example, the authors of [9] proposed a locally weighted ensemble framework to combine multiple models for transfer learning, where the weights of different models are approximated using a graph-based approach; the authors of [12] proposed a semi-supervised multi-task learning framework, where t-step transition probabilities in a Markov random walk are incorporated into the neighborhood-conditional likelihood function to find the optimal parameters. Generally speaking, none of these methods try to propagate the label information to the features irrelevant to the source domain and the unlabeled examples in the target domain via the common features. Some non-graph-based methods try to address this problem in an ad-hoc way, such as [4], whereas our paper provides a principled way to do the propagation.

The rest of the paper is organized as follows. Firstly, Section 2 introduces the tripartite graph and a simple iterative algorithm for transfer learning based on this graph. Then in Section 3, we present the graph-based transfer learning framework and associate it with the iterative algorithm from Section 2. Experimental results are shown in Section 4, followed by some discussion. Section 5 introduces related work. Finally, we conclude the paper in Section 6.

2. TRANSFER LEARNING WITH TRIPARTITE GRAPH

In this section, we first introduce the tripartite graph that propagates the label information from the source domain to the target domain via the features. Using this graph, we can obtain a classification function that takes values on all the unlabeled examples from the target domain. Then we present an iterative algorithm to find the classification function efficiently.

2.1 Notation

Let $\mathcal{X}^S$ denote the set of examples from the source domain, i.e. $\mathcal{X}^S = \{x_1^S, \ldots, x_m^S\} \subset \mathbb{R}^d$, where m is the number of examples from the source domain, and d is the dimensionality of the feature space. Let $\mathcal{Y}^S$ denote the labels of these examples, i.e. $\mathcal{Y}^S = \{y_1^S, \ldots, y_m^S\} \subset \{-1, 1\}^m$, where y_i^S is the class label of x_i^S, $1 \leq i \leq m$. Similarly, for the target domain, let $\mathcal{X}^T$ denote the set of examples, i.e. $\mathcal{X}^T = \{x_1^T, \ldots, x_n^T\} \subset \mathbb{R}^d$, where n is the number of examples from the target domain. Among these examples, only the first ϵn examples are labeled, i.e. $\mathcal{Y}^T = \{y_1^T, \ldots, y_{\epsilon n}^T\} \subset \{-1, 1\}^{\epsilon n}$, where y_i^T is the class label of x_i^T, $1 \leq i \leq \epsilon n$. Here $0 \leq \epsilon \ll 1$, i.e. only a small fraction of the examples in the target domain are labeled, and $\epsilon = 0$ corresponds to no labeled examples in the target domain. Our goal is to find a classification function for all the unlabeled examples in $\mathcal{X}^T$ with a small error rate.

2.2 Tripartite Graph

Let $G^{(3)} = \{V^{(3)}, E^{(3)}\}$ denote the undirected tripartite graph, where $V^{(3)}$ is the set of nodes in the graph, and $E^{(3)}$ is the set of weighted edges. $V^{(3)}$ consists of three types of nodes: the labeled nodes, i.e. the nodes that correspond to the labeled examples (most of them are from the source domain); the feature nodes, i.e. the nodes that correspond to the features; and the unlabeled nodes, i.e. the nodes that correspond to the unlabeled examples from the target domain. Both the labeled nodes and the unlabeled nodes are connected to the feature nodes, but the labeled nodes are not connected to the unlabeled nodes, and the nodes of the same type are not connected either. Furthermore, there is an edge between a labeled (unlabeled) node and a feature node if and only if the corresponding example has that feature, i.e. $x_{i,j}^S \neq 0$ ($x_{i,j}^T \neq 0$), where $x_{i,j}^S$ ($x_{i,j}^T$) is the j^{th} feature component of x_i^S (x_i^T), and the edge weight is set to $x_{i,j}^S$ ($x_{i,j}^T$). Here we assume that the edge weights are non-negative. This is true in many applications, such as document analysis where each feature corresponds to a unique word and the edge weight is binary or equal to the tfidf value. In a general setting, this may not be the case. However, we could perform a linear transformation to the features and make them non-negative.

Fig. 1a shows an example of the tripartite graph. The diamond-shaped nodes correspond to the feature nodes, the lighter circle nodes correspond to the examples from the source domain, and the darker circle nodes correspond to the examples from the target domain. Notice that the labeled nodes are on the left hand side, the feature nodes are in the middle, and the unlabeled nodes are on the right hand side. The intuition of the graph can be explained as follows. Consider sentiment classification in different domains as an example. Each of the diamond-shaped nodes in Fig. 1a corresponds to a unique word; the lighter circle nodes correspond to labeled movie reviews; and the darker circle nodes correspond to product reviews that we are interested in. The labeled reviews on the left hand side of Fig. 1a propagate their label information to the unlabeled product reviews via the feature nodes. Notice that each of the two domains may have some unique words that never occur in the other domain. For example, the word 'actor' often occurs in a movie review, but may never occur in a product review; similarly, the word 'polyethylene' may occur in a product review, but is never seen in a movie review. Based on this graph structure, the label information can be propagated to the domain-specific words, i.e. the words irrelevant to the movie reviews, which will help classify the unlabeled product reviews.

Given the tripartite graph, we define affinity matrix $A^{(3)}$, which is $(m+n+d) \times (m+n+d)$. The first $m+\epsilon n$ rows (columns) correspond to the labeled nodes, the next $n-\epsilon n$ rows (columns) correspond to the unlabeled nodes, and the remaining d rows (columns) correspond to the feature nodes. Therefore, $A^{(3)}$ has the following block structure

$$A^{(3)} = \begin{bmatrix} 0_{(m+\epsilon n)\times(m+\epsilon n)} & 0_{(m+\epsilon n)\times(n-\epsilon n)} & A^{(3,1)} \\ 0_{(n-\epsilon n)\times(m+\epsilon n)} & 0_{(n-\epsilon n)\times(n-\epsilon n)} & A^{(3,2)} \\ (A^{(3,1)})^T & (A^{(3,2)})^T & 0_{d\times d} \end{bmatrix}$$

where $0_{a\times b}$ is an $a \times b$ 0 matrix, $A^{(3,1)}$ and $A^{(3,2)}$ are both sub-matrices of $A^{(3)}$, and $(\cdot)^T$ is the transpose of a matrix. Let $A_{i,j}^{(3,1)}$ ($A_{i,j}^{(3,2)}$) denote the element in the i^{th} row and the j^{th} column of $A^{(3,1)}$ ($A^{(3,2)}$). Based on the discussion above, $A_{i,j}^{(3,1)} = x_{i,j}^S$ and $A_{i,j}^{(3,2)} = x_{i,j}^T$. Note that the elements of $A^{(3)}$ are non-negative. Furthermore, define diagonal matrix $D^{(3)}$, which is $(m+n+d) \times (m+n+d)$. Its diagonal element $D_i^{(3)} = \sum_{j=1}^{m+n+d} A_{i,j}^{(3)}$, $i = 1, \ldots, m+n+d$, where $A_{i,j}^{(3)}$ denote the element in the i^{th} row and the j^{th} column of $A^{(3)}$. Similar as $A^{(3)}$, $D^{(3)}$ has the following block structure

$$D^{(3)} = \begin{bmatrix} D^{(3,1)} & 0_{(m+\epsilon n)\times(n-\epsilon n)} & 0_{(m+\epsilon n)\times d} \\ 0_{(n-\epsilon n)\times(m+\epsilon n)} & D^{(3,2)} & 0_{(n-\epsilon n)\times d} \\ 0_{d\times(m+\epsilon n)} & 0_{d\times(n-\epsilon n)} & D^{(3,3)} \end{bmatrix}$$

where $D^{(3,1)}$, $D^{(3,2)}$ and $D^{(3,3)}$ are diagonal matrices whose diagonal elements are equal to the row sums of $A^{(3,1)}$, $A^{(3,2)}$ and $(A^{(3,1)})^T + (A^{(3,2)})^T$ respectively. Finally, define the normalized affinity matrix $S^{(3)} = (D^{(3)})^{-1/2} A^{(3)} (D^{(3)})^{-1/2}$, which also has the following block structure

$$S^{(3)} = \begin{bmatrix} 0_{(m+\epsilon n)\times(m+\epsilon n)} & 0_{(m+\epsilon n)\times(n-\epsilon n)} & S^{(3,1)} \\ 0_{(n-\epsilon n)\times(m+\epsilon n)} & 0_{(n-\epsilon n)\times(n-\epsilon n)} & S^{(3,2)} \\ (S^{(3,1)})^T & (S^{(3,2)})^T & 0_{d\times d} \end{bmatrix}$$

where $S^{(3,1)} = (D^{(3,1)})^{-\frac{1}{2}} A^{(3,1)} (D^{(3,3)})^{-\frac{1}{2}}$, and $S^{(3,2)} = (D^{(3,2)})^{-\frac{1}{2}} A^{(3,2)} (D^{(3,3)})^{-\frac{1}{2}}$. Similar as $A^{(3)}$, the elements of $S^{(3)}$ are also non-negative.

2.3 Objective Function Q_1

Given the tripartite graph and the corresponding affinity matrix, we can define three functions f^L, f^F and f^U, which take values on the labeled nodes, the feature nodes, and the unlabeled nodes respectively. Note that the function value of f^U will be used to classify the unlabeled examples in the target domain, and the function value of f^F can be used to infer the polarity of the features. Similarly, define three vectors y^L, y^F and y^U, whose lengths are equal to the number of labeled nodes $m + \epsilon n$, the number of feature nodes d, and the number of unlabeled nodes $n - \epsilon n$ respectively. The elements of y^L are set to be the class label of the corresponding labeled example, whereas the elements of y^F and y^U could reflect our prior knowledge about the polarity of the features and the unlabeled examples, or simply 0 if such information is not available. For the sake of notation simplicity, let $f = [(f^L)^T, (f^U)^T, (f^F)^T]^T$ and $y = [(y^L)^T, (y^U)^T, (y^F)^T]^T$.

To find the classification function with a low error rate, we propose to minimize the following objective function, which is motivated by [25].

$$Q_1(f) = \frac{1}{2} \sum_{i,j=1}^{m+n+d} A_{ij}^{(3)} \left(\frac{f_i}{\sqrt{D_i^{(3)}}} - \frac{f_j}{\sqrt{D_j^{(3)}}}\right)^2 + \mu \sum_{i=1}^{m+n+d} (f_i - y_i)^2$$
$$= f^T (I_{(m+n+d)\times(m+n+d)} - S^{(3)}) f + \mu \|f - y\|^2$$

where μ is a small positive parameter, $I_{a\times b}$ is an $a \times b$ identity matrix, and f_i and y_i are the i^{th} element of f and y respectively.

This objective function can be interpreted as follows. The first term of Q_1, $f^T(I_{(m+n+d)\times(m+n+d)} - S^{(3)})f$, measures the label smoothness of f. In other words, neighboring nodes on the graph should have similar f values. The second term, $\mu\|f - y\|^2$, measures the consistency of f with the label information and the prior knowledge encoded in y. By minimizing Q_1, we hope to obtain a smooth classification function f_U with a small error rate.

In our implementation, we fix $f^L = y^L$. In this way, we can make better use of the label information in y^L. This modification distinguishes our method from the manifold ranking algorithm proposed in [25], where each element of f needs to be optimized. Minimizing Q_1 with the above constraint, we have the following lemma.

Lemma 1. *If $f^L = y^L$, Q_1 is minimized at*

$$f^{U*} = (I_{(n-\epsilon n)\times(n-\epsilon n)} - \alpha^2 S^{(3,2)} (S^{(3,2)})^T)^{-1} \tag{1}$$
$$((1-\alpha)y^U + \alpha(1-\alpha) S^{(3,2)} y^F + \alpha^2 S^{(3,2)} (S^{(3,1)})^T y^L)$$
$$f^{F*} = (I_{d\times d} - \alpha^2 (S^{(3,2)})^T S^{(3,2)})^{-1} ((1-\alpha) y^F \tag{2}$$
$$+ \alpha (S^{(3,1)})^T y^L + \alpha(1-\alpha)(S^{(3,2)})^T y^U)$$

where $\alpha = \frac{1}{1+\mu}$.

Proof. Replacing f^L with y^L, Q_1 becomes

$$Q_1 = (y^L)^T y^L + (f^U)^T f^U + (f^F)^T f^F - 2(y^L)^T S^{(3,1)} f^F$$
$$- 2(f^U)^T S^{(3,2)} f^F + \mu\|f^U - y^U\|^2 + \mu\|f^F - y^F\|^2$$

Therefore,

$$\frac{\partial Q_1}{\partial f^U} = 2f^U - 2S^{(3,2)} f^F + 2\mu(f^U - y^U)$$
$$\frac{\partial Q_1}{\partial f^F} = 2f^F - 2(S^{(3,1)})^T y^L - 2(S^{(3,2)})^T f^U + 2\mu(f^F - y^F)$$

Setting $\frac{\partial Q_1}{\partial f^U}$ and $\frac{\partial Q_1}{\partial f^F}$ to 0, we get Equations 1 and 2. $\square$

Notice that in Lemma 1, in order to get f^{U*} and f^{F*}, we need to solve matrix inversions. This is computationally expensive especially when the number of unlabeled examples in $\mathcal{X}^T$ or the number of features is very large. To address this problem, we propose the following iteration steps to obtain the optimal solutions.

$$f^U(t+1) = \alpha S^{(3,2)} f^F(t) + (1-\alpha) y^U \tag{3}$$

$$f^F(t+1) = \alpha (S^{(3,1)})^T y^L + \alpha (S^{(3,2)})^T f^U(t) + (1-\alpha) y^F \tag{4}$$

where $f^U(t)$ and $f^F(t)$ denote f^U and f^F at the t^{th} iteration. The two equations can be interpreted as follows. Based on Equation 3, if an example has many positive (negative) features or it is believed to be positive (negative) a priori, its function value would be large (small), indicating that it is a positive (negative) example. Based on Equation 4, if a feature is contained in many positive (negative) labeled examples, or it is shared by many unlabeled examples with large (small) function values, or it is believed to be positive (negative) a priori, its function value would be large (small). In this way, the label information is gradually propagated to the unlabeled examples in the target domain and the features irrelevant to the source domain via the common features on the tripartite graph.

The following theorem guarantees the convergence of the iteration steps.

Theorem 1. *When t goes to infinity, $f^U(t)$ converges to f^{U*} and $f^F(t)$ converges to f^{F*}.*

Proof. According to Equations 3 and 4

$$f^U(t) = \alpha S^{(3,2)} f^F(t-1) + (1-\alpha) y^U = (1-\alpha) y^U$$
$$+ \alpha S^{(3,2)} (\alpha (S^{(3,1)})^T y^L + \alpha (S^{(3,2)})^T f^U(t-2)$$
$$+ (1-\alpha) y^F)$$
$$= \alpha^2 S^{(3,2)} (S^{(3,2)})^T f^U(t-2) + (1-\alpha) y^U$$
$$+ \alpha^2 S^{(3,2)} (S^{(3,1)})^T y^L + \alpha(1-\alpha) S^{(3,2)} y^F$$

For the sake of simplicity, let $V = S^{(3,2)}(S^{(3,2)})^T$ and $v = (1-\alpha)y^U + \alpha^2 S^{(3,2)}(S^{(3,1)})^T y^L + \alpha(1-\alpha)S^{(3,2)}y^F$. First, we assume that t is an even number. Therefore, the above equation can be written as follows.

$$f^U(t) = \alpha^2 V f^U(t-2) + v = (\alpha^2 V)^{\frac{t}{2}} f^U(0) + (\sum_{i=0}^{\frac{t}{2}-1} (\alpha^2 V)^i)v$$

where $f^U(0)$ is the initial value of f^U. Since $\alpha = \frac{1}{1+\mu}$, $0 < \alpha < 1$. Therefore, if the eigenvalues of V are in [-1,1], we have

$$\lim_{t\to\infty} (\alpha^2 V)^{\frac{t}{2}} f^U(0) = 0_{(n-\epsilon n)\times 1}$$

$$\lim_{t\to\infty} \sum_{i=0}^{\frac{t}{2}-1} (\alpha^2 V)^i = (I_{(n-\epsilon n)\times(n-\epsilon n)} - \alpha^2 V)^{-1}$$

Hence, if t is an even number,

$$\lim_{t\to\infty} f^U(t) = f^{U*}$$

With respect to the eigenvalues of V, we have the following lemma.

Lemma 2. *The eigenvalues of V are in [-1,1].*

Proof. Notice that $V = S^{(3,2)}(S^{(3,2)})^T = (D^{(3,2)})^{-\frac{1}{2}} A^{(3,2)} (D^{(3,3)})^{-\frac{1}{2}} (D^{(3,3)})^{-\frac{1}{2}} (A^{(3,2)})^T (D^{(3,2)})^{-\frac{1}{2}}$ is similar to $(D^{(3,2)})^{-1} A^{(3,2)} (D^{(3,3)})^{-1} (A^{(3,2)})^T$. Let $V^{(1)} = (D^{(3,2)})^{-1} A^{(3,2)}$ and $V^{(2)} = A^{(3,2)}(D^{(3,3)})^{-1}$. Then V is similar to $V^{(1)}(V^{(2)})^T$. Furthermore, it is easy to see that $\sum_{j=1}^{d} V^{(1)}_{i,j} = 1$, $\forall 1 \le i \le n - \epsilon n$, and $\sum_{i=1}^{n-\epsilon n} V^{(2)}_{i,j} = 1$, $\forall 1 \le j \le d$, where $V^{(1)}_{i,j}$ and $V^{(2)}_{i,j}$ are the elements of $V^{(1)}$ and $V^{(2)}$ in the ith row and the jth column. Therefore, for the ith row of $V^{(1)}(V^{(2)})^T$, $\forall 1 \le i \le n - \epsilon n$, its row sum is

$$\sum_{j=1}^{n-\epsilon n} \sum_{k=1}^{d} V^{(1)}_{i,k} V^{(2)}_{j,k} = \sum_{k=1}^{d} V^{(1)}_{i,k} \sum_{j=1}^{n-\epsilon n} V^{(2)}_{j,k} = 1$$

According to Perron-Frobenius theorem [18], since the elements of $V^{(1)}(V^{(2)})^T$ are non-negative, the spectral radius of $V^{(1)}(V^{(2)})^T$ is 1. Furthermore, since V is similar to $V^{(1)}(V^{(2)})^T$, its spectral radius is also 1. Therefore, the eigenvalues of V are in [-1,1]. □

Therefore, using Lemma 2, we have shown that if t is an even number, as t goes to infinity, $f^U(t)$ converges to f^{U*}. This conclusion also holds when t is an odd number. Finally, applying similar techniques to f^F, we can show that as t goes to infinity, $f^F(t)$ converges to f^{F*}.

Comparing the above iterative steps with Equations 1 and 2, we can see that they avoid solving matrix inversions directly. In our experiments, the number of iteration steps until convergence is always less than 30. Therefore, these iterative steps are an efficient alternative to Equations 1 and 2.

Based on Equations 3 and 4, we design the following **TRITER** (TRIpartite-graph-based TransfER learning) algorithm to minimize Q_1, which is shown in Algorithm 1. It works as follows. First, we set $y^L(f^L)$, y^U and y^F according to the label information or our prior knowledge. $f^U(0)$ and $f^F(0)$ are initialized to y^U and y^F respectively. Next, we update f^U and f^F according to Equations 3 and 4. Finally, we classify all the unlabeled examples in $\mathcal{X}^T$ according to the corresponding elements in f^U.

Algorithm 1 TRITER Algorithm for Transfer Learning

```
Input: The set of examples from the source domain $\mathcal{X}^S$ and
    the set of their labels $\mathcal{Y}^S$; the set of examples from the
    target domain $\mathcal{X}^T$ and the set of labels for the first $\epsilon n$
    examples $\mathcal{Y}^T$; the number of iteration steps $t$; $\mu$.
Output: The labels of all the unlabeled examples in $\mathcal{X}^T$.
1: Set $y^L(f^L)$ according to the label information; set $y^U$
   and $y^F$ according to our prior knowledge, or simply 0 if
   such information is not available; initialize $f^U(0) = y^U$
   and $f^F(0) = y^F$.
2: for $i = 1 : t$ do
3:    Calculate $f^U(i)$ and $f^F(i)$ according to Equations 3
      and 4.
4: end for
5: for $i = (\epsilon n + 1) : n$ do
6:    If the function value of $f^U(t)$ at $x_i^T$ is positive, $y_i^T = 1$;
      otherwise, $y_i^T = -1$.
7: end for
```

3. GRAPH-BASED TRANSFER LEARNING FRAMEWORK

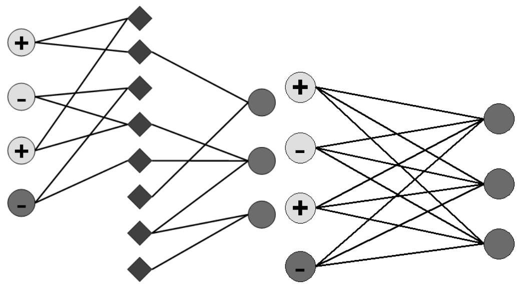
In Section 2, we have introduced a tripartite graph that connects the examples from the source domain and the target domain with the features, and have designed the **TRITER** algorithm to minimize the objective function Q_1 efficiently. Although simple and straight-forward, Q_1 is not best suited for transfer learning. This is because the label information from the source domain and the target domain is propagated in the same way. If the labeled examples from the source domain dominate the labeled nodes, the label information of the small number of labeled examples from the target domain would be flooded, and the resulting classification function for the target domain may be largely biased. In other words, since our goal is to construct an accurate classifier in the target domain, the labeled examples from the same domain should be more important than the labeled examples from different domains.

To address this problem, in this section, we propose the graph-based transfer learning framework. In this framework, in addition to the tripartite graph, we also design a bipartite graph to make better use of the labeled examples from the target domain. Based on the two graphs, we present objective function Q_2 and the optimal solutions. Furthermore, under certain conditions, the solutions to Q_2 can be obtained by minimizing a slightly modified version of Q_1 via the **TRITER** algorithm.

3.1 Bipartite Graph

Let $G^{(2)} = \{V^{(2)}, E^{(2)}\}$ denote the undirected bipartite graph, where $V^{(2)}$ is the set of nodes in the graph, and $E^{(2)}$ is the set of weighted edges. $V^{(2)}$ consists of two types of nodes: the labeled nodes which correspond to the labeled examples from both the source domain (majority) and the target domain (minority); the unlabeled nodes which correspond to the unlabeled examples from the target domain. Each labeled node is connected to each unlabeled node, with the edge weight indicating the domain related similarity between the two examples, whereas the same type of nodes are not connected.

Fig. 1b shows an example of the bipartite graph which has the same labeled and unlabeled nodes as in Fig. 1a. Similarly, the lighter circle nodes correspond to the examples from the source domain, and the darker circle nodes correspond to the examples from the target domain. The labeled nodes on the left hand side are connected to each unlabeled node on the right hand side. Again take sentiment classification in different domains as an example. The labeled nodes correspond to all the labeled reviews, most of which are movie reviews, and the unlabeled nodes correspond to all the unlabeled product reviews. The edge weights are set to reflect the domain related similarity between two reviews. Therefore, if two reviews are both product reviews, one labeled and one unlabeled, their edge weight would be large; whereas if two reviews are from different domains, the movie review labeled and the product review unlabeled, their edge weight would be small. In this way, we hope to make better use of the labeled product reviews to construct the classification function for the unlabeled product reviews.



(a) Tripartite graph (b) Bipartite graph

Figure 1: An example of the graphs.

Let $A^{(2)}$ denote the affinity matrix for the bipartite graph, which is $(m+n)\times(m+n)$. The first $m+\epsilon n$ rows (columns) correspond to the labeled nodes, and the remaining $n-\epsilon n$ rows (columns) correspond to the unlabeled nodes. According to the structure of the bipartite graph, $A^{(2)}$ has the following form.

$$A^{(2)} = \begin{bmatrix} 0_{(m+\epsilon n)\times(m+\epsilon n)} & A^{(2,1)} \\ (A^{(2,1)})^T & 0_{(n-\epsilon n)\times(n-\epsilon n)} \end{bmatrix}$$

where $A^{(2,1)}$ is the sub-matrix of $A^{(2)}$. Note that the elements of $A^{(2)}$ are set to be non-negative. Let $D^{(2)}$ denote the $(m+n)\times(m+n)$ diagonal matrix, the i^{th} diagonal element of which is defined $D_i^{(2)} = \sum_{j=1}^{m+n} A_{i,j}^{(2)}$, $i = 1,\ldots,m+n$, where $A_{i,j}^{(2)}$ is the element of $A^{(2)}$ in the i^{th} row and the j^{th} column. Similar as $A^{(2)}$, $D^{(2)}$ has the following block structure.

$$D^{(2)} = \begin{bmatrix} D^{(2,1)} & 0_{(m+\epsilon n)\times(n-\epsilon n)} \\ 0_{(n-\epsilon n)\times(m+\epsilon n)} & D^{(2,2)} \end{bmatrix}$$

where $D^{(2,1)}$ and $D^{(2,2)}$ are diagonal matrices whose diagonal elements are equal to the row sums and the column sums of $A^{(2,1)}$ respectively. Finally, let $S^{(2)}$ denote the normalized affinity matrix $S^{(2)} = (D^{(2)})^{-1/2}A^{(2)}(D^{(2)})^{-1/2}$, which also has the following block structure.

$$S^{(2)} = \begin{bmatrix} 0_{(m+\epsilon n)\times(m+\epsilon n)} & S^{(2,1)} \\ (S^{(2,1)})^T & 0_{(n-\epsilon n)\times(n-\epsilon n)} \end{bmatrix}$$

where $S^{(2,1)} = (D^{(2,1)})^{-\frac{1}{2}}A^{(2,1)}(D^{(2,2)})^{-\frac{1}{2}}$.

3.2 Objective Function Q_2

In Subsection 2.2, we introduced a tripartite graph which propagates the label information from the labeled nodes to the unlabeled nodes via the feature nodes; and in Subsection 3.1, we introduced a bipartite graph which puts high weights on the edges connecting examples from the same domain and low weights on the edges connecting examples from different domains. In this section, we combine the two graphs to design objective function Q_2. By minimizing Q_2, we can obtain a smooth classification function for the unlabeled examples in the target domain which relies more on the labeled examples from the target domain than on those from the source domain.

For the sake of simplicity, define $g = [(f^L)^T, (f^U)^T]^T$. It is easy to see that $g = Bf$, where $B = [I_{(m+n)\times(m+n)}, 0_{(m+n)\times d}]$. Thus the objective function Q_2 can be written as follows.

$$\begin{aligned} Q_2(f) &= \frac{1}{2}\gamma \sum_{i,j=1}^{m+n+d} A_{i,j}^{(3)}\left(\frac{f_i}{\sqrt{D_i^{(3)}}} - \frac{f_j}{\sqrt{D_j^{(3)}}}\right)^2 \\ &+ \frac{1}{2}\tau \sum_{i,j=1}^{m+n} A_{i,j}^{(2)}\left(\frac{g_i}{\sqrt{D_i^{(2)}}} - \frac{g_j}{\sqrt{D_j^{(2)}}}\right)^2 + \mu \sum_{i=1}^{m+n+d}(f_i - y_i)^2 \\ &= \gamma f^T(I_{(m+n+d)\times(m+n+d)} - S^{(3)})f \\ &+ \tau f^T B^T(I_{(m+n)\times(m+n)} - S^{(2)})Bf + \mu\|f-y\|^2 \end{aligned}$$

where γ and τ are two positive parameters. Similar as in Q_1, the first term of Q_2, $\gamma f^T(I_{(m+n+d)\times(m+n+d)} - S^{(3)})f$, measures the label smoothness of f on the tripartite graph; the second term, $\tau f^T B^T(I_{(m+n)\times(m+n)} - S^{(2)})Bf$, measures the label smoothness of f on the bipartite graph; and the third term, $\mu\|f-y\|^2$, measures the consistency of f with the label information and the prior knowledge. It should be pointed out that the first two terms in Q_2 can be combined mathematically; however, the two graphs can not be combined due to the normalization process.

Based on Q_2, we can claim that our method is different from semi-supervised learning, which treats the labeled examples from different domains in the same way. In our method, by imposing the label smoothness constraint on the bipartite graph, we can see that the labeled examples from the target domain have more impact on the unlabeled examples from the same domain than the labeled examples from the source domain. In the next section, we will compare our method with a semi-supervised learning method experimentally.

Similar as before, we fix $f^L = y^L$, and minimize Q_2 with respect to f^U and f^F. The solutions can be obtained by the following lemma.

LEMMA 3. *If $f^L = y^L$, Q_2 is minimized at*

$$\tilde{f}^{U*} = ((\gamma+\tau+\mu)I_{(n-\epsilon n)\times(n-\epsilon n)} - \frac{\gamma^2}{\gamma+\mu}S^{(3,2)}(S^{(3,2)})^T)^{-1} \quad (5)$$

$$(\mu y^U + \frac{\gamma^2}{\gamma+\mu}S^{(3,2)}(S^{(3,1)})^T y^L + \frac{\gamma\mu}{\gamma+\mu}S^{(3,2)}y^F + \tau(S^{(2,1)})^T y^L)$$

$$\tilde{f}^{F*} = \frac{\gamma}{\gamma+\mu}((S^{(3,1)})^T y^L + (S^{(3,2)})^T \tilde{f}^{U*}) + \frac{\mu}{\gamma+\mu}y^F \quad (6)$$

PROOF. Replacing f^L with y^L, Q_2 becomes

$$\begin{aligned} Q_2 &= \gamma((y^L)^T y^L + (f^U)^T f^U + (f^F)^T f^F \\ &- 2(y^L)^T S^{(3,1)} f^F - 2(f^U)^T S^{(3,2)} f^F) \\ &+ \tau((y^L)^T y^L + (f^U)^T f^U - 2(y^L)^T S^{(2,1)} f^U) \\ &+ \mu\|f^U - y^U\|^2 + \mu\|f^F - y^F\|^2 \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{\partial Q_2}{\partial f^U} &= 2\gamma(f^U - S^{(3,2)}f^F) + 2\tau(f^U - (S^{(2,1)})^T y^L) \\ &+ 2\mu(f^U - y^U) \\ \frac{\partial Q_2}{\partial f^F} &= 2\gamma(f^F - (S^{(3,1)})^T y^L - (S^{(3,2)})^T f^U) \\ &+ 2\mu(f^F - y^F) \end{aligned}$$

Setting $\frac{\partial Q_2}{\partial f^U}$ and $\frac{\partial Q_2}{\partial f^F}$ to 0, we get Equations 5 and 6. □

In Equation 5, if we ignore the matrix inversion term in the front, we can see that $\tilde{f}^{U*}$ gets the label information from the labeled nodes through the following two terms: $\frac{\gamma^2}{\gamma+\mu}S^{(3,2)}(S^{(3,1)})^T y^L$ and $\tau(S^{(2,1)})^T y^L$, which come from the tripartite graph and the bipartite graph respectively. Recall that y^L is defined on the labeled nodes from both the source domain and the target domain. In particular, if a labeled node is from the target domain, its corresponding row in $S^{2,1}$ would have large values, and it will make a big contribution to $\tilde{f}^{U*}$ via $\tau(S^{(2,1)})^T y^L$. This is in contrast to labeled nodes from the source domain, whose corresponding rows in $S^{2,1}$ have small values, and their contribution to $\tilde{f}^{U*}$ would be small as well.

Similar to objective function Q_1, we can also design an iterative algorithm to find the solutions of Q_2. However, in the following, we focus on the relationship between Q_1 and Q_2, and will introduce an iterative algorithm based on the **TRITER** algorithm to solve Q_2.

Comparing Equations 1 with 5, we can see that they are very similar to each other. The following theorem builds a connection between objective functions Q_1 and Q_2.

THEOREM 2. *If $f^L = y^L$, then $\tilde{f}^{U*}$ can be obtained by minimizing Q_1 with the following parametrization*

$$\begin{aligned} \alpha' &= \frac{\gamma}{\sqrt{(\mu+\gamma)(\mu+\gamma+\tau)}} \\ y'^L &= y^L \\ y'^U &= \frac{\mu y^U + \tau(S^{(2,1)})^T y^L}{\mu+\gamma+\tau-\gamma\sqrt{\frac{\mu+\gamma+\tau}{\mu+\gamma}}} \\ y'^F &= \frac{\mu}{\sqrt{(\mu+\gamma)(\mu+\gamma+\tau)}-\gamma}y^F \end{aligned}$$

PROOF. Replacing α, y^L, y^U and y^F with α', y'^L, y'^U and y'^F respectively in Equations 1, we get Equations 5. □

The most significant difference between the parameter settings in Theorem 2 and the original settings is in the definition of y'^U. That is, y'^U consists of two parts, one from its own prior information, which is in proportion to μy^U, and the other from the label information of the labeled examples, which is in proportion to $\tau(S^{(2,1)})^T y^L$. Note that the second part is obtained via the bipartite graph, and it encodes the domain information. In other words, incorporating the bipartite graph into the transfer learning framework is equivalent to working with the tripartite graph alone, with a domain specific prior for the unlabeled examples in the target domain and slightly modified versions of α and y^F.

Finally, to minimize Q_2, we can simply apply the **TRITER** algorithm with the parameter settings specified in Theorem 2, which usually converges within 30 iteration steps.

4. EXPERIMENTAL RESULTS

In this section, we present some experimental results, and compare the proposed graph-based transfer learning framework with state-of-the-art techniques.

4.1 Experiment Settings

To demonstrate the performance of the proposed graph-based transfer learning framework, we perform experiments in the following 3 areas.

1. Sentiment classification (SC). In this area, we use the movie and product review data set. The movie reviews come from [15]. Positive labels are assigned to ratings above 3.5 stars and negative to 2 and fewer stars. The product reviews are collected from Amazon for software worth more than 50 dollars. In our experiments, we use the movie reviews as the source domain and the product reviews as the target domain. After stemming and stop word removal, the feature space is 34305-dimensional.

2. Document classification (DC). In this area, we use the 20 newsgroups data set [16]. The documents within this data set has a two-level categorical structure. Based on this structure, we generate 3 transfer learning tasks. Each task involves distinguishing two higher-level categories. The source domain and the target domain contains examples from different lower-level categories. For example, one transfer learning task is to distinguish between rec and talk. The source domain contains examples from rec.sport.baseball and talk.politics.misc; whereas the target domain contains examples from rec.sport.hockey and talk.religion.misc. The way

that the transfer learning tasks are generated is similar to [9] and [6]. After stemming and stop word removal, the feature space is 53975-dimensional.

3. Intrusion detection (ID). In this area, we use the KDD Cup 99 data set [1]. It consists of both normal connections and attacks of different types, including DOS (denial-of-service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to local superuser privileges), and probing (surveillance and other probing). For this data set, we also generate 3 transfer learning tasks. In each task, both the source domain and the target domain contain some normal examples as the positive class, but the negative class in the two domains corresponds to different types of attacks. Similar as in [9], only the 34 continuous features are used.

The details of the transfer learning tasks are summarized in Table 1. Notice that in SC and DC, we tried both binary features and tfidf features. It turns out that binary features lead to better performance. Therefore, we only report the experimental results with the binary features here. Note that the features in ID are not binary.

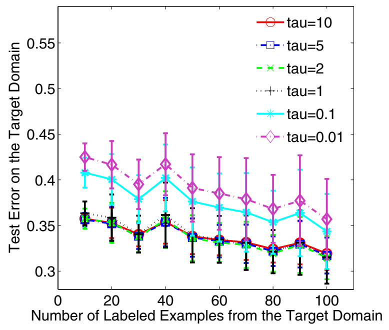
In our proposed transfer learning framework, the bipartite graph is constructed as follows. $A^{(2,1)}$ is a linear combination of two matrices. The first matrix is based on domain information, i.e. its element is set to 1 iff the corresponding labeled and unlabeled examples are both from the target domain, and it is set to 0 otherwise. The second matrix is $A^{(3,1)}(A^{(3,2)})^T$, i.e. if a labeled example shares a lot of features with an unlabeled example, the corresponding element in this matrix is large. Note that this is only one way of constructing the bipartite graph with domain information. Exploring the optimal bipartite graph for transfer learning is beyond the scope of this paper.

We compare our method with the following methods.

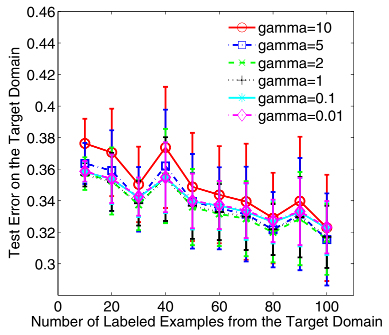
1. Learning from the target domain only, which is denoted **target only**. With this method, we ignore the source domain, and construct the classification function solely based on the labeled examples from the target domain. In other words, none of the nodes in the tripartite graph and bipartite graph correspond to examples from the source domain.
2. Learning from the source domain only, which is denoted **source only**. With this method, we ignore the label information from the target domain, and construct the classification function solely based on the labeled examples from the source domain. In other words, all of the nodes on the left hand side of the tripartite graph and the bipartite graph correspond to examples from the source domain, and the nodes that correspond to the target domain examples are all on the right hand side of the two graphs.
3. Learning from both the source domain and the target domain, which is denoted **source+target**. With this method, we combine the function f^U output by **target only** and **source only** linearly, and predict the class labels of the unlabeled examples accordingly.
4. Semi-supervised learning, denoted **semi-supervised**. It is based on the manifold ranking algorithm [25]. With this method, all the labeled examples are considered from the target domain, and we propagate their label information to the unlabeled examples in the same way.
5. The transfer learning toolkit developed by UC Berkeley (http://multitask.cs.berkeley.edu/). The method that we use is based on [2], which is denoted **BTL**. Note that for document classification and sentiment classification, the feature space is too large to be processed by **BTL**. Therefore, as a preprocessing step, we perform singular value decomposition (SVD) to project the data onto the 100-dimensional space spanned by the first 100 singular vectors.
6. The boosting-based transfer learning method [7], which is denoted **TBoost**.

4.2 Evaluations

For the graph-based transfer learning framework, we set $\mu = 0.01$, which is consistent with [25], $y^F = 0$, and $y^U = 0$ in all the experiments. For τ and γ, we test their impact on the performance using SC, which is shown in Fig. 2. From this figure, we can see that the performance of our method is quite stable within a wide range of τ and γ. Therefore, in the following experiments, we set $\tau = 5$ and $\gamma = 1$.



(a) $\gamma = 1$



(b) $\tau = 5$

Figure 2: Impact of τ and γ on the performance of the proposed method.

Fig. 3 to Fig. 9 compare the proposed graph-based transfer learning framework with the baseline methods on the 7

Table 1: Transfer Learning Tasks

Area	Source Positive	Source Negative	Target Positive	Target Negative
SC	Movie (1000)	Movie (1000)	Product (5680)	Product (6047)
DC	comp.os.ms-windows .misc (572)	rec.autos (592)	comp.windows.x (592)	rec.motorcycles (596)
	rec.sport.baseball (594)	sci.crypt (594)	rec.sport.hockey (598)	sci.electronics (591)
	rec.sport.baseball (594)	talk.politics.misc (464)	rec.sport.hockey (598)	talk.religion.misc (376)
ID	Normal (1000)	Probing (1000)	Normal (1000)	R2L (1000)
	Normal (1000)	DOS (1000)	Normal (1000)	R2L (1000)
	Normal (1000)	Probing (500) DOS (500)	Normal (1000)	R2L (1000)

transfer learning tasks. In these figures, the x-axis is the number of labeled examples from the target domain, and the y-axis is the average test error in the target domain over 20 runs (labeled examples from the target domain are randomly picked in each run). The error bars are also shown in these figures.

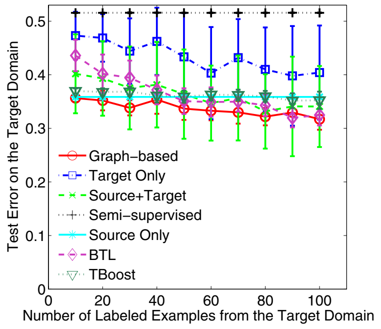


Figure 3: Comparison on SC.

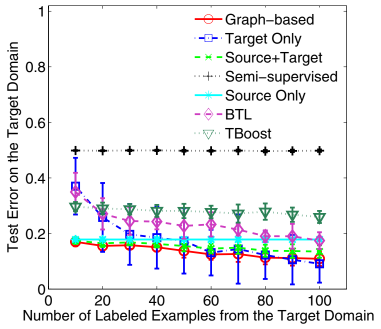


Figure 4: Comparison on the first task of DC.

Based on these results, we have the following observations. First of all, it is easy to see that our graph-based method is the best of the 7 methods in all the tasks in terms of the average error rate. Second, the graph-based method is very stable in terms of the small error bars, especially compared with **target only**. This is consistent with our intuition since **target only** totally ignores the source domain, and only uses the label information from the target domain to

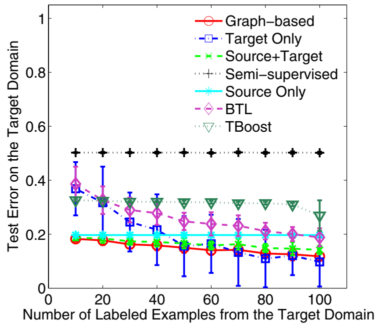


Figure 5: Comparison on the second task of DC.

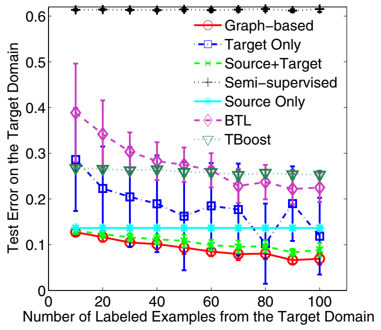


Figure 6: Comparison on the third task of DC.

construct the classification function. When the number of labeled examples from the target domain is small, its performance varies a lot depending on the specific labeled examples. In contrast, the graph-based method considers the label information from both the source domain and the target domain, therefore, it is not very sensitive to the specific labeled examples from the target domain. Third, the performance of **semi-supervised** is always much worse than our method. This is because in all our experiments, the number of labeled examples from the target domain is much smaller than that from the source domain, which is quite common in practise. Therefore, with **semi-supervised**, the labeled examples from the target domain is flooded by those from the source domain, and the performance is not satisfactory. Fourth, in most of the experiments, the average performance

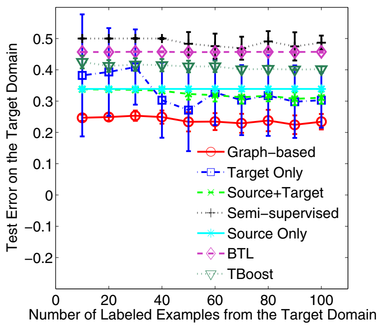


Figure 7: Comparison on the first task of ID.

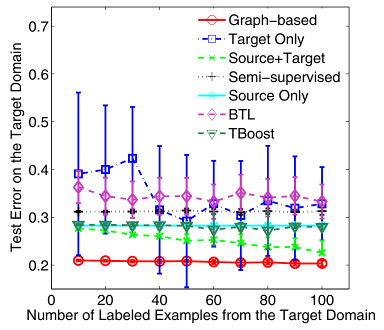


Figure 8: Comparison on the second task of ID.

of the graph-based method and **target only** is getting closer as we increase the number of labeled examples from the target domain. This is because with the graph-based method, the labeled examples from the target domain have more impact on the classification function than those from the source domain. As the number of labeled examples from the target domain increases, their impact tends to dominate. So the performance of the graph-based method and **target only** will get closer. Finally, in some experiments, such as Fig. 4 and Fig. 6, the gap between the graph-based method and **source+target** is getter larger. This is reasonable since in **source+target**, we are combining the source domain and the target domain in a naive way. So the performance gain caused by more labeled examples from the target domain is not as significant as the graph-based method.

5. RELATED WORK

There has been significant amount of work on transfer learning in machine learning research. One of the early attempts aims to achieve better generalization performance by jointly modeling multiple related learning tasks, and transferring information among them, i.e. multi-task learning [3, 5, 19]. It usually tackles the problem where the feature space and the feature distribution $P(x)$ are identical whereas the labeling functions are different. Further developments in the area include combining labeled data from the source domain with labeled or unlabeled data from the target domain, which leads to transfer learning methods for k-nearest neighbor [19], support vector machines [21], and logistic regression [11]. Another line of research focuses on Bayesian logistic regression with a Gaussian prior on the parameters [2,

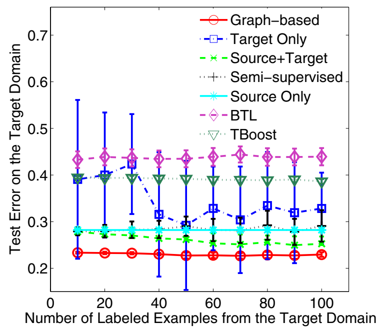


Figure 9: Comparison on the third task of ID.

10]. There are also specialized transfer learning techniques for certain application areas, such as adapting context-free grammar [17], speech recognition [13], and sentiment prediction [4].

Transfer learning is closely related to concept drifting in stream mining, in which the statistical properties of the target variable change over time. These changing properties might be the class prior $P(y)$, the feature distribution $P(x|y)$, the decision function $P(y|x)$ or a combination of all. Multiple approaches have been developed, such as ensemble approaches [20], co-clustering [6], and local structure map [9]. Transfer learning is also relevant to sample bias correction, which is mostly concerned with distinct training distribution $P(x|\lambda)$ and testing distribution $P(x|\theta)$ with unknown parameters λ and θ. Several bias correction methods have been developed based on estimating the probability that an example is selected into the sample and using rejection sampling to obtain unbiased samples of the correct distribution [23, 22, 8].

Our proposed framework is motivated by the graph-based methods for semi-supervised learning [26, 25]. In the framework, the tripartite graph propagates the label information from the source domain to the target domain via the features, and the bipartite graph makes better use of the label information from the target domain. This framework is fundamentally different from previous work on transfer learning and related areas. It propagates the label information in a principled way, which is in contrast to some ad-hoc methods based on pivot features [4]; it directly associates the polarity of features with the class labels of all the examples, which is in contrast to previous graph-based methods [12, 9] that do not model this relationship with the graph structure.

6. CONCLUSION

In this paper, we proposed a new graph-based framework for transfer learning. It is based on both a tripartite graph and a bipartite graph. The tripartite graph consists of three types of nodes, and it propagates the label information via the features. The bipartite graph consists of two types of nodes, and it imposes the domain related smoothness constraint between the labeled examples and the unlabeled examples. Based on the two graphs, we have designed an objective function Q_2, which is a weighted combination of the label smoothness on the tripartite graph, the label smoothness on the bipartite graph, and the consistency with the label information and the prior knowledge. Closed form solutions to Q_2 have been developed. Furthermore, we have built the connection between Q_2 and the objective function Q_1, which is solely based on the tripartite graph. Finally, based on the above connection, we have designed an iterative algorithm to find the solutions to Q_2. Different from existing transfer learning methods, the proposed framework propagates the label information to both the features irrelevant to the source domain and the unlabeled examples from the target domain via the common features in a principled way. Experimental results on several transfer learning tasks demonstrate the superiority of the proposed framework over state-of-the-art techniques. For future work, we are interested in investigating the theoretical bounds of the performance for graph-based transfer learning algorithms and the applications to large-scale data sets.

7. REFERENCES

[1] Kdd cup 99. In *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*, 1999.

[2] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

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

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

[5] R. Caruana. Multitask learning. In *Machine Learning*, pages 41–75, 1997.

[6] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *KDD*, pages 210–219, 2007.

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

[8] W. Fan, I. Davidson, B. Zadrozny, and P. S. Yu. An improved categorization of classifier's sensitivity on sample selection bias. In *ICDM*, pages 605–608, Washington, DC, USA, 2005. IEEE Computer Society.

[9] J. Gao, W. Fan, J. Jiang, and J. Han. Knowledge transfer via multiple model local structure mapping. In *KDD*, pages 283–291, 2008.

[10] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *ICML*, pages 489–496, 2007.

[11] X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. In *ICML*, pages 505–512, 2005.

[12] Q. Liu, X. Liao, and L. Carin. Semi-supervised multitask learning. In *NIPS*, 2007.

[13] J. luc Gauvain and C. hui Lee. Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE Transactions on Speech and Audio Processing*, 2:291–298, 1994.

[14] A. M. Martínez. Recognition of partially occluded and/or imprecisely localized faces using a probabilistic approach. In *CVPR*, pages 1712–1717, 2000.

[15] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *CoRR*, cs.CL/0205070, 2002.

[16] J. Rennie. 20 newsgroups. In *http://people.csail.mit.edu/jrennie/20Newsgroups/*, 2007.

[17] B. Roark and M. Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *NAACL*, pages 126–133, 2003.

[18] H. Roger and J. Charles. *Matrix Analysis*. Cambridge University Press, 1985.

[19] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, pages 640–646. MIT Press, 1996.

[20] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *KDD '03*, 2003.

[21] P. Wu and T. G. Dietterich. Improving svm accuracy by training on auxiliary data sources. In *ICML*, pages 871–878, 2004.

[22] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML*, page 114, 2004.

[23] B. Zadrozny and C. Elkan. Learning and making decisions when costs and probabilities are both unknown. In *KDD*, pages 204–213, New York, NY, USA, 2001. ACM.

[24] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *NIPS*, 2005.

[25] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *NIPS*, 2003.

[26] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.