

Metric Learning for Graph-based Domain Adaptation

*Paramveer S. Dhillon*¹, *Partha Talukdar*², *Koby Crammer*³

¹CIS, University of Pennsylvania, U.S.A

²MLD, Carnegie Mellon University, U.S.A

³EE, The Technion, Israel

dhillon@cis.upenn.edu, partha.talukdar@cs.cmu.edu,

koby@ee.technion.ac.il

Abstract

In many domain adaption formulations, it is assumed to have large amount of unlabeled data from the domain of interest (target domain), some portion of it may be labeled, and large amount of labeled data from other domains, aka source domain(s). Motivated by the fact that labeled data is hard to obtain in any domain, we design algorithms for the settings in which there exists large amount of unlabeled data from all domains, small portion of which may be labeled.

We build on recent advances in graph-based semi-supervised learning and supervised metric learning. Given all instances, labeled and unlabeled, from all domains, we build a large similarity graph between them, where an edge exists between two instances if they are close according to some metric. Instead of using predefined metric, as commonly performed, we feed the labeled instances into metric-learning algorithms and (re)construct a data-dependent metric, which is used to construct the graph. We employ different types of edges depending on the domain-identity of the two vertices touching it, and learn the weights of each edge.

We provide extensive empirical evidence demonstrating that our approach leads to significant reduction in classification error across domains, and evaluate the contribution of each resource: labeled and unlabeled data of the various domains.

Keywords: Sentiment Analysis, Machine Learning, Domain Adaptation.

1 Introduction

Domain adaptation is an important machine learning subtask where the goal is to perform well on a particular classification task on a *target domain*, especially when most of the resources are available from other different domains, called *source(s) domain(s)* (Pan and Yang, 2009), and only limited amount of supervision is available to the target domain. In the standard setting, most domain adaptation algorithms assume the availability of large amounts of labeled data for the source domain, with little or no labeled data from the target domain (Arnold et al., 2008; Dai et al., 2007; Wang et al., 2009). However, in many practical situations, obtaining labeled data from *any* domain is expensive and time consuming, while unlabeled data is easily available. This setting of domain adaptation, where there is only limited amount of labeled data and large amounts of unlabeled data, both from all domains, is relatively unexplored.

To address the issue of labeled data sparsity even within a single domain, recent research has focused on Semi-Supervised Learning (SSL) algorithms, which learn from limited amounts of labeled data combined with widely available unlabeled data. Examples of a few graph-based SSL algorithms

include Gaussian Random Fields (GRF) (Zhu et al., 2003), Quadratic Criteria (QR) (Bengio et al., 2006). Given a set of instances that contain small amount of labeled instances and a majority that is unlabeled, most graph based SSL algorithms first construct a graph where each node corresponds to an instance. Similar nodes are connected by an edge, with edge weight encoding the degree of similarity. Once the graph is constructed, the nodes corresponding to labeled instances are injected with the corresponding label. Using this initial label information along with the graph structure, graph based SSL algorithms assign labels to all unlabeled nodes in the graph. Most of the graph based SSL algorithms are iterative and also parallelizable, making them suitable for large scale SSL setting where vast amounts of unlabeled data is usually available.

Most of the graph based SSL algorithms mentioned above concentrate primarily on the label inference part, i.e. assigning labels to nodes *once the graph has already been constructed*, with very little emphasis on construction of the graph itself. Only recently, the issue of graph construction has begun to receive attention (Wang and Zhang, 2006; Jebara et al., 2009; Daitch et al., 2009). Most of these methods emphasize on constructing graphs which satisfy certain structural properties (e.g. degree constraints on each node). Since our focus is on SSL, a certain number of labeled instances are available at our disposal. However, the graph construction methods mentioned above are all unsupervised in nature, i.e. they do not utilize available label information during the graph construction process. As recently proposed by (Dhillon et al., 2010), the available label information can be used to *learn* a distance metric, which can then be used to set the edge weights in the constructed graph.

In this paper, we bring together these three lines of work: domain adaptation, graph-based SSL, and metric learning for graph construction, and make the following contributions:

1. We consider an important setting for domain adaptation: one where most of the data is unlabeled and only limited amount of instances are labeled. This holds across *all* domains. This setting is relatively unexplored.
2. To the best of our knowledge, we are the first to employ graph-based non-parametric methods for domain adaptation.
3. We provide extensive experimental results on real-world datasets, to both demonstrate the effectiveness of metric learning for graph construction for domain adaptation, and analyze the contributions of each component.

2 Related Work

Several methods for domain adaptation have recently been proposed (Arnold et al., 2008; Blitzer et al., 2006; Dai et al., 2007; Pan and Yang, 2009; Eaton et al., 2008; Wang et al., 2009). In (Arnold et al., 2008), the labeled data comes entirely from the source domain, while certain amount of unlabeled target data is also used during transduction. Similar setting is also explored in (Dai et al., 2007; Wang et al., 2009). In contrast to these methods, we assume that limited amount of labeled data and large amounts of unlabeled data from both source and target domains are available. This is motivated by the fact that obtaining large amount of labeled data from any domain is expensive to prepare. The method presented by (Blitzer et al., 2006) also explores a similar setting, but our method is easier to implement and it does not make use of the high domain specific prior knowledge (i.e., for pivot selection) performed by (Blitzer et al., 2006).

All previously proposed methods mentioned above are parametric in nature. The graph-based adaptation method presented in this paper is non-parametric. To the best of our knowledge, it is novel in the context of domain adaptation. The method of (Wang et al., 2009) is similar in spirit as both employ graphs, yet they use a hybrid graph structure involving both instances and features for transfer learning, while we focus on domain adaptation and use homogeneous graph consisting of instance nodes only. Another important difference is that the graphs their algorithms build do *not* take available label information into account, while our algorithms do take such information into account. We will see below in Section 8, that this leads to significant improvement in performance. Another work similar in spirit to ours is of (Eaton et al., 2008). They build a graph over tasks (i.e., a node in such a graph is a task) to decide on the transferability among different tasks for transfer learning. In contrast, we focus on domain adaptation and build a graph over data instances, i.e., a node in our graph corresponds to a data instance.

A method for transfer learning via dimensionality reduction, Maximum Mean Discrepancy Embedding (MMDE), is presented in (Pan et al., 2008). MMDE explores a setting where only source labeled and target unlabeled data are available. This is different from the setting considered in this paper where unlabeled and limited amounts of labeled data are available from both source and target domains. Moreover, MMDE minimizes the divergence between *marginal* distributions between source and target domains. This is different from the metric learning-based objective optimized in this paper.

Most domain adaptation methods assume that while the data generation probability changes from one domain to another, the predictive distribution, i.e., the probability of label given data instance, remains constant across domains. Unfortunately, this is not always true in many real world settings. In order to address this problem, Predictive Distribution Matching SVM (PDM-SVM) has been recently proposed (Seah et al., 2010). PDM-SVM exploits multiple source domains to identify high density regions in which the source class labels are same as the target class labels. PDM-SVM estimates predictive distribution of the target domain by using an iterative transduction procedure, where at each iteration, a Laplacian-SVM style regularization (Belkin et al., 2006) is used to automatically label unlabeled target instances, which are then added to the training pool in the next iteration. During each transductive step, a kNN graph is constructed. There are significant differences between PDM-SVM and IGC (the method proposed in this paper): (1) PDM-SVM is parametric while IGC is non-parametric; (2) the graph used by PDM-SVM doesn't allow an edge between two instances from the same domain while the graph used in IGC does. In spite of these differences, the two methods are complimentary: IGC can be used to construct a metric learning-based task-dependent graph to be used as the regularizer in PDM-SVM, as opposed to the current unsupervised graph used by PDM-SVM.

3 Notation

We denote by n_l^s and n_u^s , the number of labeled and unlabeled instances (respectively) from the source domain. Similarly, n_l^t and n_u^t are the number of labeled and unlabeled instances from the target domain. Denote by n the total number of instances. Let X be the $d \times n$ matrix of d -dimensional column instances (from source and target domains combined). We define the $n \times n$ diagonal label-indicator matrix S to be $S_{ii} = 1$ iff instance x_i is labeled, and zero otherwise. We denote by \mathcal{L} the set of all possible labels of size $m = |\mathcal{L}|$. We define the $n \times m$ instance-label matrix by Y , where $Y_{i,j} = 1$ iff the i th instance is labeled by the j th label. Note, that the i th column of Y is undefined if $S_{i,i} = 0$, i.e. the data instance is not labeled. Similarly, we denote by \hat{Y} the $n \times m$ matrix of estimated label information, i.e. output of a inference algorithm (e.g. see Section

5). Such algorithms assign a labeling score to all instances, including labeled and unlabeled.

4 Domain Adaptation

Formally, we consider the following problem. Given, a total of $n_l^s + n_l^t$ labeled instances from the source(s) and target domains combined, and in addition $n_u^s + n_u^t$ unlabeled instances from the same domains. Our goal is to label these n_u^t unlabeled instances from the target domain (domain of interest). The task is challenging and non-trivial since we assume that $n_l^s \ll n_u^s$, and similarly $n_l^t \ll n_u^t$. Our setting is different from previous approaches in two ways: First, we assume small amount of labeled data from all domains, as opposed to most previous work in domain adaption which have focused in the ‘‘asymmetric’’ case where there is large amount of labeled source instances, and only very few, if any, labeled target instances. Second, we compensate, this lack in labeled data by considering unlabeled data from all domains, source and target, as opposed to previous settings which assumed unlabeled data only from the target domain. We believe that our ‘‘symmetric’’ setting is very realistic, since labeled data is expensive in any domain.

In Section 8, we report the results of experiments using a sentiment dataset, which contains reviews on products from a few categories. We assume that only a few instances are hand-labeled with the correct sentiment for every category, and our goal is to exploit the labeled and *unlabeled* instances from all domains to perform well on a single pre-defined *target* domain. Our task is harder, since we have only few labeled examples from each domain, however, we exploit additional cheap resource, namely unlabeled data from all the domains.

5 Graph Construction & Inference

Given a set X of n instances, both from the source and target domains, we construct a graph where each instance is associated with a node. We add an edge between two nodes if the two nodes are similar and the edge’s weight represents the degree of similarity between the corresponding instances. Denote the resulting graph by $G = (V, E, W)$ be this graph, where $V = V_l^s \cup V_u^s \cup V_l^t \cup V_u^t$ is the set of vertices with $|V| = n$, $|V_l^s| = n_l^s$, $|V_u^s| = n_u^s$, $|V_l^t| = n_l^t$, $|V_u^t| = n_u^t$; E is the set of edges, and W is the symmetric $n \times n$ matrix of edge weights. W_{ij} is the weight of edge (i, j) which is monotonic in the similarity between instances x_i and x_j . Additionally, $V^s = V_l^s \cup V_u^s$, and $V^t = V_l^t \cup V_u^t$ are the set of vertices associated with sources and target domain instances, respectively. Gaussian kernel (Zhu et al., 2003) is a widely used measure of similarity between data instances, which can be used to compute edge weights as shown in Eq. (1).

$$W_{ij} = \alpha_{ij} \times \exp\left(-d_A(x_i, x_j)/(2\sigma^2)\right) \quad (1)$$

where $d_A(x_i, x_j)$ is the distance measure between instances x_i and x_j and A is a positive definite matrix of size $d \times d$, which parameterizes the (squared) Mahalanobis distance (Eq. (3)). Furthermore, σ is the kernel bandwidth parameter, and $\alpha_{ij} = \alpha$ ($0 \leq \alpha \leq 1$) if the edge connects instances from two different domains, and $\alpha_{ij} = 1$, otherwise. In other words, the hyperparameter, α , controls the importance of cross domain edges. Setting edge weights directly using Eq. (1) results in a complete graph, where *any* two pair of nodes are connected, since the Gaussian kernel always attains strictly positive values by definition. This is undesirable as the graph is dense (and in fact complete) and thus all computation times are at least quadratic in the number of instances, which may be very large. We thus generate a sparse graph by retaining only edges to k nearest neighbors of each node, and dropping all other edges (i.e. setting corresponding edge weights to 0), a commonly used graph sparsification strategy. The number of edges in the resulting graph is linear in the number of instances.

With the graph $G = (V, E, W)$ constructed, we perform inference over this graph to assign labels to all n_u unlabeled nodes. This is done by propagating the label information from the labeled nodes to the unlabeled nodes. Any of the several graph based SSL algorithms mentioned in Section 1 may be used for this task. For the experiments in this paper, we use the GRF algorithm (Zhu et al., 2003) which minimizes the optimization problem shown in (2).

$$\min_{\hat{Y}} \sum_{i,j} \sum_{l \in \mathcal{L}} W_{i,j} (\hat{Y}_{il} - \hat{Y}_{jl})^2, \quad \text{s.t. } SY = S\hat{Y} \quad (2)$$

As outlined in (Zhu et al., 2003), this optimization can be efficiently and exactly solved to obtain \hat{Y} . The result, is a labeling of all instances, including the n_u^t unlabeled instances from the target domain.

In most previous graph-based SSL methods (e.g. (Zhu et al., 2003)), the matrix A is predefined to the identity $A = I$, in Eq. (1), resulting in the standard Euclidean distance in input space. This method of unsupervised graph construction is *not* task dependent. Instead, we also learn the matrix A using the (small) set of labeled instances using metric learning algorithms. We add more detail below in Section 7. In a nutshell, we construct a similarity metric tailored to the current specific adaptation task.

6 Metric Learning Review

We now review a recently proposed *supervised* method for learning Mahalanobis distance between instance pairs. We shall concentrate on learning the PSD matrix $A \succeq 0$ which parametrizes the distance, $d_A(x_i, x_j)$, between instances x_i and x_j .

$$d_A(x_i, x_j) = (x_i - x_j)^\top A (x_i - x_j) \quad (3)$$

This is equivalent to finding a linear transformation P of the input space, and then applying Euclidean distance on the transformed instances Px_i .

Information-Theoretic Metric Learning (ITML) (Davis et al., 2007) assumes the availability of prior knowledge about inter-instance distances. In this scheme, similar instances should have low Mahalanobis distance between them, i.e., $d_A(x_i, x_j) \leq u$, for some non-trivial upper bound u . Similarly, dissimilar instances should have a large distance between them, that is, $d_A(x_i, x_j) \geq l$ for some l . Given a set of similar instances S and dissimilar instances D , the ITML algorithm chooses the matrix A that minimizes the following optimization problem:

$$\begin{aligned} \min_{A \succeq 0, \xi} \quad & D_{\text{Id}}(A, A_0) + \gamma \cdot D_{\text{Id}}(\xi, \xi_0) \\ \text{s.t.} \quad & \text{tr}\{A(x_i - x_j)(x_i - x_j)^\top\} \leq \xi_{c(i,j)}, \quad \forall (i, j) \in S \\ & \text{tr}\{A(x_i - x_j)(x_i - x_j)^\top\} \geq \xi_{c(i,j)}, \quad \forall (i, j) \in D \end{aligned} \quad (4)$$

where γ is a hyperparameter which determines the importance of violated constraints and A_0 is a Mahalanobis matrix provided using prior knowledge. To solve the optimization problem in (4), an algorithm involving repeated Bregman projections is presented in (Davis et al., 2007), which we use for the experiments reported in this paper.

7 Using Labeled Data for Graph Construction

We now describe how to incorporate labeled and unlabeled data during graph construction. We start with a review of a new graph construction framework (Dhillon et al., 2010) which combines existing

Algorithm 1 Supervised Graph Construction (SGC) **Input:** instances X , training labels Y , training instance indicator S , neighborhood size k **Output:** Graph edge weight matrix, W

- 1: $A \leftarrow \text{MetricLearner}(X, S, Y)$
 - 2: $W \leftarrow \text{ConstructKnnGraph}(X, A, k)$
 - 3: return W
-

Algorithm 2 Iterative Graph Construction (IGC) **Input:** instances X , training labels Y , training instance indicator S , label entropy threshold β , neighborhood size k **Output:** Graph edge weight matrix, W

- 1: $\hat{Y} \leftarrow Y, \hat{S} \leftarrow S$
 - 2: **repeat**
 - 3: $W \leftarrow \text{SGC}(X, \hat{Y}, k)$
 - 4: $\hat{Y}' \leftarrow \text{GraphLabelInference}(W, \hat{S}, \hat{Y})$
 - 5: $U \leftarrow \text{SelectLowEntInstances}(\hat{Y}', \hat{S}, \beta)$
 - 6: $\hat{Y} \leftarrow \hat{Y} + U\hat{Y}'$
 - 7: $\hat{S} \leftarrow \hat{S} + U$
 - 8: **until** convergence (i.e., $U_{ii} = 0, \forall i$)
 - 9: return W
-

supervised metric learning algorithms (such as ITML) with *transductive* graph-based label inference to learn a new distance metric from labeled as well as unlabeled data combined. In self-training styled iterations, IGC alternates between graph construction and label inference; with output of label inference used during next round of graph construction, and so on.

7.1 Iterative Graph Construction (IGC)

IGC builds on the assumption that supervised (metric) learning improves with more labeled data. Since we are focusing on the SSL setting with n_l labeled and n_u unlabeled instances, the algorithm automatically labels the unlabeled instances using some existing graph based SSL algorithm, and then includes a subset of the labeled instances in the training set for the next round of metric learning. Naturally, only examples with low assigned label entropy (i.e., high confidence label assignments) are used. Specifically, we use a threshold parameter $\beta > 0$ to determine which examples will be used for the next round. (In practice we set $\beta = 0.05$ and observed that indeed most of the low entropy instances which are selected for inclusion in next iteration of metric learning, are classified correctly.) This iterative process continues until no new instances are set of labeled instances. This occurs when either all the instances are already exhausted, or when none of the remaining unlabeled instances can be assigned labels with high confidence.

The IGC framework is presented in Algorithm 2. The algorithm iterates between the two main steps as follows. In Line 1, any supervised metric learner, such as ITML, may be used as the MetricLearner. Using the distance metric learned in Line 1, a new k-NN graph is constructed in Line 2, whose edge weight matrix is stored in W . In Line 4, GraphLabelInference optimizes over the newly constructed graph the GRF objective (Zhu et al., 2003) shown in Eq. (5).

$$\min_{\hat{Y}'} \text{tr}\{\hat{Y}'^T L \hat{Y}'\}, \text{ s.t. } \hat{S}\hat{Y} = \hat{S}\hat{Y}' \quad (5)$$

where $L = D - W$ is the (unnormalized) Laplacian, and D is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$.

Source-Target Domain Pairs	Source Only Labeled		Both Source & Target Labeled		Target Only
	$n_l^s = 100$ $n_l^t = 0$	$n_l^s = 200$ $n_l^t = 0$	$n_l^s = 100$ $n_l^t = 100$	$n_l^s = 200$ $n_l^t = 100$	$n_l^s = 0$ $n_l^t = 100$
Electronics-DVDs	48.3 ± 0.2	48.1 ± 0.1	45.1 ± 0.2	41.4 ± 0.2	45.9 ± 0.5
DVDs-Electronics	43.2 ± 0.3	43.2 ± 0.2	38.3 ± 0.3	36.5 ± 0.3	39.1 ± 0.7
DVDs-Books	47.1 ± 0.4	46.6 ± 0.3	42.5 ± 0.3	40.3 ± 0.4	43.3 ± 0.4
Books-DVDs	48.3 ± 0.3	48.1 ± 0.3	45.6 ± 0.2	43.1 ± 0.3	45.9 ± 0.5
Music-Books	46.9 ± 0.2	46.6 ± 0.2	42.1 ± 0.4	39.9 ± 0.3	43.3 ± 0.4
Books-Music	41.0 ± 0.3	40.7 ± 0.1	38.9 ± 0.3	36.4 ± 0.2	39.1 ± 0.7
Video-Electronics	43.1 ± 0.3	43.3 ± 0.2	38.1 ± 0.4	35.5 ± 0.3	39.1 ± 0.7
Electronics-Video	44.2 ± 0.2	44.0 ± 0.1	39.3 ± 0.2	36.6 ± 0.4	40.4 ± 0.3
Video-DVDs	48.1 ± 0.4	47.9 ± 0.2	44.9 ± 0.3	42.0 ± 0.3	45.9 ± 0.5
DVDs-Video	44.1 ± 0.1	43.8 ± 0.1	39.1 ± 0.1	36.8 ± 0.2	40.4 ± 0.3
Kitchen-Apparel	42.3 ± 0.4	42.1 ± 0.3	36.5 ± 0.1	33.8 ± 0.3	37.1 ± 0.4
Apparel-Kitchen	44.9 ± 0.3	44.5 ± 0.2	39.4 ± 0.3	37.0 ± 0.4	41.0 ± 0.7

Table 1: Domain adaptation classification errors (lower is better) using GRF (see Section 5) in the PCA space. Total instances $n = 3000$. All target unlabeled instances ($n_u^t = 1400$) are used for evaluation, with results averaged over four trials. See text for details.

The constraint, $\hat{S}\hat{Y} = \hat{S}\hat{Y}'$, in (5) makes sure that labels on training instances are not changed during inference. In Line 5, a currently unlabeled instance x_i (i.e., $\hat{S}_{ii} = 0$) is considered a new labeled training instance, i.e. $U_{ii} = 1$, for next round of metric learning if the instance has been assigned labels with high confidence in the current iteration, i.e., if its label distribution has low entropy (i.e., $\text{Entropy}(\hat{Y}'_{i'}) \leq \beta$). Finally in Line 6, training instance label information is updated. This iterative process is continued till no new labeled instance can be added, i.e., when $U_{ii} = 0 \forall i$. IGC returns the learned matrix A which can be used to compute Mahalanobis distance using Eq. (3). The number of parameters estimated by IGC (i.e., dimensions of W) increases as the number data instances increase. Hence, we note that that IGC is non-parametric, just as other graph-based methods.

8 Experiments

Data: We use data from 12 domain pairs obtained from (Crammer et al., 2009), and preprocessed to keep only those features which occurred more than 20 times. The classification task is the following: given a product review, predict user’s sentiment, i.e., whether it is positive or negative. Hence, this is a binary classification problem with number of classes $m = 2$. A total of 1,500 instances from each domain were sampled, i.e., $n = 3000$. We note that the goal is to label unlabeled target data (n_u^t), so in all experiments reported below we have at least 1,300 instances to be labeled.

Experimental Setup: We used cosine similarity¹ (using appropriate A) to set edge weights, followed by k -NN graph sparsification, as described in Section 5. The hyperparameters $k \in \{2, 5, 10, 50, 100, 200, 500, 1000\}$ and the Gaussian kernel bandwidth multiplier², $\rho \in \{1, 2, 5, 10, 50, 100\}$, are tuned on a separate development set. The hyperparameter, α (see Eq. (1))

¹We experimented with both Gaussian kernels and cosine similarity, and cosine similarity lead to better performance, and we use it in all experiments.

² $\sigma = \rho \sigma_0$, where ρ is the tuned multiplier, and σ_0 is set to average distance.

Source-Target Domain Pairs	Source Only Labeled		Both Source & Target Labeled		Target Only
	$n_l^s = 100$ $n_l^t = 0$	$n_l^s = 200$ $n_l^t = 0$	$n_l^s = 100$ $n_l^t = 100$	$n_l^s = 200$ $n_l^t = 100$	$n_l^s = 0$ $n_l^t = 100$
Electronics-DVDs	43.3 ± 0.4	43.1 ± 0.3	40.8 ± 0.2	38.3 ± 0.3	41.7 ± 0.3
DVDs-Electronics	33.6 ± 0.2	33.4 ± 0.1	29.9 ± 0.4	27.9 ± 0.3	30.8 ± 0.3
DVDs-Books	36.3 ± 0.3	36.3 ± 0.3	32.1 ± 0.3	31.9 ± 0.4	33.3 ± 0.6
Books-DVDs	43.7 ± 0.4	43.8 ± 0.3	40.6 ± 0.3	40.3 ± 0.2	41.7 ± 0.3
Music-Books	35.9 ± 0.5	35.8 ± 0.2	32.4 ± 0.1	30.1 ± 0.3	33.3 ± 0.6
Books-Music	36.9 ± 0.2	36.7 ± 0.3	33.1 ± 0.3	31.8 ± 0.5	34.7 ± 0.4
Video-Electronics	32.9 ± 0.3	32.8 ± 0.4	29.7 ± 0.4	28.4 ± 0.3	30.8 ± 0.3
Electronics-Video	37.9 ± 0.2	37.7 ± 0.3	33.8 ± 0.3	32.9 ± 0.4	35.1 ± 0.3
Video-DVDs	43.0 ± 0.3	42.9 ± 0.1	41.0 ± 0.4	40.1 ± 0.3	41.7 ± 0.3
DVDs-Video	38.0 ± 0.4	38.0 ± 0.2	34.6 ± 0.3	33.0 ± 0.2	35.1 ± 0.3
Kitchen-Apparel	35.1 ± 0.3	35.0 ± 0.3	33.5 ± 0.4	32.9 ± 0.5	33.9 ± 0.5
Apparel-Kitchen	31.9 ± 0.4	31.8 ± 0.2	28.9 ± 0.3	27.5 ± 0.4	29.9 ± 0.3

Table 2: Domain adaptation classification errors (lower is better) using GRF (see Section 5) in the *learned metric space* (IGC). Total instances $n = 3000$. All target unlabeled instances ($n_u^t = 1400$) are used for evaluation, with results averaged over four trials. See text for details.

was tuned over the range $[0.1, 1]$, with step size 0.1. The α value which gave the best GRF objective (Eq. (2)) was selected. Please note that this is an automatic parameter selection mechanism requiring no additional held out data. For all graph-based experiments, GRF (see Section 5) is used as the inference algorithm.

Setting The Mahalanobis Matrix A: We consider two methods to set the value of the matrix A . First, instances are projected into a lower dimensional space using Principal Components Analysis (PCA). For all experiments, dimensionality of the projected space was set at 250. We set $A = P^T P$, where P is the projection matrix generated by PCA. We found the baseline algorithms to perform better in this space than the input d -dimensional space, and hence this is used as the original space. Second, the matrix A is learned by applying IGC (Algorithm 2) (see Section 7) on the PCA projected space (above); with ITML used as MetricLearner in IGC. We use standard implementations of ITML and IGC made available by respective authors.

8.1 Comparison with Unsupervised Graph Construction using PCA

Table 1 summarizes the results for the PCA based approach and Table 2 summarizes the results for the IGC approach. Each line corresponds to a single pair of source and target. The right column of both tables shows baseline test error (averaged over four runs), where the only supervision is via $n_l^t = 100$ labeled instances from the target domain. In fact, this unique setting is where there is no adaptation. Comparing the results of both algorithms, clearly the IGC algorithm outperforms the PCA approach. In other words, using the labeled data to construct a metric captures better the relation between instances from both classes, compared with a method that minimizes the variance like PCA.

Source-Target Domain Pairs	Fixed α ($\alpha = 1$)	Tuned α (best α value in bracket)
Electronics-DVDs	38.9 ± 0.2	38.3 ± 0.3 (0.8)
DVDs-Electronics	28.3 ± 0.2	27.9 ± 0.3 (0.7)
DVDs-Books	32.5 ± 0.2	31.9 ± 0.4 (0.7)
Books-DVDs	41.0 ± 0.3	40.3 ± 0.2 (0.6)
Music-Books	30.6 ± 0.2	30.1 ± 0.3 (0.6)
Books-Music	32.3 ± 0.1	31.8 ± 0.5 (0.7)
Video-Electronics	29.6 ± 0.2	28.4 ± 0.3 (0.9)
Electronics-Video	33.3 ± 0.1	32.9 ± 0.4 (0.8)
Video-DVDs	40.7 ± 0.2	40.1 ± 0.3 (0.8)
DVDs-Video	33.9 ± 0.3	33.0 ± 0.2 (0.7)
Kitchen-Apparel	33.5 ± 0.4	32.9 ± 0.5 (0.5)
Apparel-Kitchen	29.0 ± 0.3	27.5 ± 0.4 (0.4)

Table 3: Domain adaptation classification errors (lower is better) in the learned metric space (IGC), for fixed value of α (left) and learned value of α . Total $n = 3000$ instances, with $n_l^s = 200$ and $n_l^t = 100$ labeled instances in all cases. All target unlabeled instances ($n_u^t = 1400$) are used for evaluation, with results averaged over four trials.

8.2 Using Only Source Labeled Data

Next, we evaluate the relative contribution of using only source labeled data to only target labeled data. The results for the former case are summarized in the left column of both Table 1 and Table 2 (PCA and IGC). For simplicity we discuss now only the IGC results, the PCA based results are similar. The left sub-column corresponds to using only 100 labeled source instances and the right sub-column corresponds to using only 200 labeled source instances. In both cases, no labeled target data is used. Comparing the most left sub-column to the most right sub-column, we see a degradation of about 3% in performance. Interestingly, using additional 100 labeled source instances does not improve performance, as indicated by the similar values of both two left sub-columns. In other words, 100 labeled target instances are “worth” more than 200 source instances. Note that in this case we used unlabeled data from both domains, 1,500 instances from the target domain (used for evaluation), and 1,400 or 1,300 instances from the source data (for the first and second experiment, respectively).

Next, we evaluated the change in performance when we use labeled data from both domains, $n_l^t = 100$ labeled instances from target domain, and either $n_l^s = 100$ labeled instances from the source domain (middle column, left sub-column) or $n_l^s = 200$ labeled instances from the source domain (middle column, right sub-column). Comparing the results with 100 labeled source data with the case where there are only labeled target instances (middle column, left sub-column vs. right columns) we see that 100 additional source data always improve performance, compared when we have only 100 labeled target instances by about 1%. The improvement to the case when we have only 100 source instances is stronger and is about 3% on average.

Learning The Cross-Domain Edge-Weight α : An evaluation of the contribution of the tuning of the cross domain edge weight multiplier, α , appears in Table 3. In all cases, we used GRF in the IGC space, with $n_l^s = 200$ labeled instances from the source domain and $n_l^t = 100$ instances from the target domain. The left column shows the results for uniform edge-weight of 1 for all type of

Source-Target Domain Pairs	No Source Unlabeled $n_u^s = 0$	With Source Unlabeled $n_u^s = 1300$
Electronics-DVDs	39.9 ± 0.3	38.3 ± 0.3
DVDs-Electronics	28.8 ± 0.3	27.9 ± 0.3
DVDs-Books	32.4 ± 0.3	31.9 ± 0.4
Books-DVDs	41.2 ± 0.2	40.3 ± 0.2
Kitchen-Apparel	34.0 ± 0.2	32.9 ± 0.5
Apparel-Kitchen	28.6 ± 0.3	27.5 ± 0.4

Table 4: Domain adaptation classification errors in IGC space, comparing with and without source data. In all cases, we use $n_l^s = 200$ and $n_l^t = 100$ labeled instances. The reported errors are on $n_u^t = 1400$ instances, results averaged over four trials.

edges, and the right column shows the results for tuned value of α using the objective value itself. Comparing the two columns we see that indeed tuning α reduces the error by about 0.7% on average. From Table 2 we learn that typical optimal values of α are far from being 1. This is explained by the fact that while the source data reduces variance by the additional 200 labeled points it also adds also some bias. The tuned value of α balances this tradeoff optimally.

8.3 Using Source Unlabeled Data

In Table 4 we evaluate the contribution of *source* unlabeled data on six pairs (rest omitted due to lack of space). The left column shows the error when source unlabeled data is not used and the right column shows the results when 1,300 unlabeled points are used. In both cases we use 200 source labeled points and 100 target labeled points. The improvement is of about 1% error. Recall that the intuition behind graph based methods is that the data lies on a low-dimensional manifold, and so when we add source unlabeled instances, we better sample this manifold. This can be also shown from the edge density in both graphs. Where no source unlabeled data is used, the average number of neighbors for the 1,400 unlabeled target instances is 20.9, which increases to 22.7 when source unlabeled data is used. This additional 2 edges on average allows the labeling of these instances to rely on more information, which we hypothesize is the reason for the improvement in performance.

8.4 Comparison with Other Methods

Domain Pairs	TSVM	EasyAdapt	IGC
Electronics-DVDS	40.1 ± 0.2	41.0 ± 0.4	38.3 ± 0.3
Books-Music	32.7 ± 0.3	33.4 ± 0.3	31.8 ± 0.5
DVDs-Videos	33.8 ± 0.4	34.9 ± 0.4	33.0 ± 0.2
Videos-Electronics	29.7 ± 0.2	30.1 ± 0.4	28.4 ± 0.3
Kitchen-Apparel	33.9 ± 0.3	33.7 ± 0.1	32.9 ± 0.5

Table 5: Classification Errors for IGC comparison with TSVM and EasyAdapt. In all cases, we use $n_l^s = 200$ and $n_l^t = 100$ labeled instances. The reported errors are on $n_u^t = 1400$ instances, results averaged over four trials.

In previous sections we have shown the superior performance of IGC over projections learnt using PCA and standard SVM (a state-of-the-art baseline which is also the top performing algorithm in the seminal sentiment classification work of (Pang et al., 2002)). However, a comparison with

Domain Pairs	SVM	PCA	IGC
Electronics-DVDs	43.1 \pm 0.3	41.4 \pm 0.2	38.3 \pm 0.3
DVDs-Electronics	37.1 \pm 0.2	36.5 \pm 0.3	27.9 \pm 0.3
DVDs-Books	41.0 \pm 0.3	40.3 \pm 0.4	31.9 \pm 0.4
Books-DVDs	43.9 \pm 0.2	43.1 \pm 0.3	40.3 \pm 0.2
Music-Books	41.0 \pm 0.3	39.9 \pm 0.3	30.1 \pm 0.3
Books-Music	36.7 \pm 0.3	36.4 \pm 0.2	31.8 \pm 0.5
Video-Electronics	35.9 \pm 0.2	35.5 \pm 0.3	28.4 \pm 0.3
Electronics-Video	37.4 \pm 0.3	36.6 \pm 0.4	32.9 \pm 0.4
Video-DVDs	43.0 \pm 0.2	42.0 \pm 0.3	40.1 \pm 0.3
DVDs-Video	38.1 \pm 0.3	36.8 \pm 0.2	33.0 \pm 0.2
Kitchen-Apparel	35.0 \pm 0.2	33.8 \pm 0.3	32.9 \pm 0.5
Apparel-Kitchen	38.2 \pm 0.3	37.0 \pm 0.4	27.5 \pm 0.4

Table 6: Classification errors (lower is better, lowest marked in bold) comparing SVM, GRF (see Section 5) in PCA space, and GRF in IGC space. Total $n = 3000$ instances, with total 300 labeled instances ($n_l^s = 200$ and $n_l^t = 100$). The reported errors are on $n_u^t = 1400$ instances, with results averaged over 4 trials.

state-of-the-art semi-supervised learning and domain adaptation approaches was pending. So, in this section we compare the performance of IGC with TSVM (Transductive SVM)—A widely large margin transductive model which has shown state-of-the-art performance on many text classification tasks (Joachims, 1999) and EasyAdapt (Daume III, 2007) which is a state-of-the-art domain adaptation algorithm. The results are shown in Table 5.

Final Results: Finally, we compared the best combination for both IGC and PCA (with source unlabeled data, 200 source labeled instances, and 100 target labeled instances) to the performance of a Support Vector Machine (SVM) classifier trained over the 300 training instances (200 from the source domain, and 100 from the target domain) using a polynomial kernel whose degree is tuned on a development set.

The results are summarized in Table 6. Clearly, for all domain pairs, GRF in PCA space is either comparable or better than SVM. This may not be surprising since SVM did not use the additional 1,300 source unlabeled data. Also, as already seen above, GRF in IGC space outperforms both SVM baseline and GRF in PCA space. This demonstrates the benefit of using a learned metric (in this case using IGC) during graph construction for graph-based domain adaptation.

9 Conclusion

We brought together three active directions of research: domain adaptation, graph-based learning, and metric learning, and made the following contributions: (1) investigated usage of unlabeled data from all domains and limited labeled data from all domains; (2) employed graph-based non-parametric methods for domain adaptation; (3) evaluated systematically the contribution of each learning resource to the final performance. We plan to further investigate improved usage of graph-based techniques to adaptation. Here, we considered only two domains at once. We plan to extend these methods for multiple source domains.

References

- Arnold, A., Nallapati, R., and Cohen, W. (2008). A comparative study of methods for transductive transfer learning. In *ICDM Workshop on Mining and Management of Biological Data.*, pages 77–82. IEEE.
- Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434.
- Bengio, Y., Delalleau, O., and Le Roux, N. (2006). Label propagation and quadratic criterion. *Semi-supervised learning*, pages 193–216.
- Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.
- Crammer, K., Dredze, M., and Kulesza, A. (2009). Multi-Class Confidence Weighted Algorithms.
- Dai, W., Xue, G., Yang, Q., and Yu, Y. (2007). Co-clustering based classification for out-of-domain documents. In *KDD*, pages 210–219. ACM.
- Daitch, S., Kelner, J., and Spielman, D. (2009). Fitting a graph to vector data. In *ICML*.
- Daume III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.
- Davis, J., Kulis, B., Jain, P., Sra, S., and Dhillon, I. (2007). Information-theoretic metric learning. In *ICML*.
- Dhillon, P., Talukdar, P., and Crammer, K. (2010). Inference-driven metric learning for graph construction. Technical report, Technical Report MS-CIS-10-18, CIS Department, University of Pennsylvania, May.
- Eaton, E., Desjardins, M., and Lane, T. (2008). Modeling transfer relationships between learning tasks for improved inductive transfer. *Machine Learning and Knowledge Discovery in Databases*, pages 317–332.
- Jebara, T., Wang, J., and Chang, S. (2009). Graph construction and b-matching for semi-supervised learning. In *ICML*.
- Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99*, pages 200–209, San Francisco, CA, USA.
- Pan, S., Kwok, J., and Yang, Q. (2008). Transfer learning via dimensionality reduction. In *Proceedings of the 23rd national conference on Artificial intelligence*, pages 677–682.
- Pan, S. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 79–86. Association for Computational Linguistics.

Seah, C., Tsang, I., Ong, Y., and Lee, K. (2010). Predictive distribution matching SVM for multi-domain learning. *Machine Learning and Knowledge Discovery in Databases*, pages 231–247.

Wang, F. and Zhang, C. (2006). Label propagation through linear neighborhoods. In *ICML*.

Wang, Z., Song, Y., and Zhang, C. (2009). Knowledge transfer on hybrid graph. In *IJCAI*, pages 1291–1296.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*.