A Graph Embedding Framework for Maximum Mean Discrepancy-Based Domain Adaptation Algorithms

Yiming Chen, Shiji Song, *Senior Member, IEEE*, Shuang Li, and Cheng Wu

***Abstract*— Domain adaptation aims to deal with learning problems in which the labeled training data and unlabeled testing data are differently distributed. Maximum mean discrepancy (MMD), as a distribution distance measure, is minimized in various domain adaptation algorithms for eliminating domain divergence. We analyze empirical MMD from the point of view of graph embedding. It is discovered from the MMD intrinsic graph that, when the empirical MMD is minimized, the compactness within each domain and each class is simultaneously reduced. Therefore, points from different classes may mutually overlap, leading to unsatisfactory classification results. To deal with this issue, we present a graph embedding framework with intrinsic and penalty graphs for MMD-based domain adaptation algorithms. In the framework, we revise the intrinsic graph of MMD-based algorithms such that the within-class scatter is minimized, and thus, the new features are discriminative. Two strategies are proposed. Based on the strategies, we instantiate the framework by exploiting four models. Each model has a penalty graph characterizing certain similarity property that should be avoided. Comprehensive experiments on visual cross-domain benchmark datasets demonstrate that the proposed models can greatly enhance the classification performance compared with the state-of-the-art methods.**

***Index Terms*— Domain adaptation, maximum mean discrepancy, graph embedding.**

I. Introduction

COMMON unsupervised machine learning strategies assume that the training data and testing data are drawn from the same distribution. In real-world applications, however, the labeled training data are often expensive or time-consuming to collect. Under such circumstance, abundant well-labeled data from other distributions, which are referred to as source domains, are expected to be leveraged. Domain adaptation aims to deal with the problem of adapting a classification or regression model learned from source data to unlabeled target domain following a different distribution.

Manuscript received October 8, 2018; revised April 24, 2019 and July 8, 2019; accepted July 8, 2019. Date of publication July 19, 2019; date of current version September 12, 2019. This work was supported by the National Key Research and Development Program of China under Grant 2016YFB1200203. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lucio Marcenaro. *(Corresponding author: Shuang Li.)*

Y. Chen, S. Song, and C. Wu are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: chenyimi15@mails.tsinghua.edu.cn; shijis@mail.tsinghua.edu.cn; wuc@tsinghua.edu.cn).

S. Li is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: shuangli@bit.edu.cn).

Digital Object Identifier 10.1109/TIP.2019.2928630

One key strategy of most domain adaptation algorithms is to force the source and target distributions to get closer to each other [26], [28], [29], [34]. Based on this strategy, Maximum Mean Discrepancy (MMD) [20] is frequently used. MMD is a popular non-parametric distribution distance metric. It measures the distance between two distributions by the maximum mean function value difference in the unit ball of a universal reproducing kernel Hilbert space. In many domain adaptation algorithms, including feature representation learning [8], [24], [35], instance reweighting [23] and classifier adaptation [13], the square of an empirical estimate of the MMD is minimized for eliminating source and target domain divergence. Particularly, the MMD terms conditioned on predicted labels are taken advantage of such that the conditional distributions within each class are also matched respectively [22], [27], [29].

In this paper, we make some further analysis on MMD metric from the point of view of *graph embedding* to study some inconspicuous properties. Graph embedding is a process aimed to find the feature representations that best fit the similarity relationship characterized by the graph $G = \{V,\ W\}$, where V is the set of vertices representing the data points, and W is a weight matrix measuring similarities between vertices. We notice that the minimization of square empirical MMD can be considered as a direct or kernel graph embedding on the learned subspace of an intrinsic graph G. In the graph G of MMD minimization, weights between points of the *same* domain are negative, indicating that these points are deemed to be dissimilar *enemies* by each other. This is opposite to the truth. As a result, during MMD minimization, points within the same domain are pulled farther from each other. In other words, the within-domain compactness is sacrificed for the distributions being closer. Furthermore, when the class-wise conditional MMD terms are utilized, the corresponding graphs will consider the *same-label* points as *enemies*, and the within-class compactness will not be well preserved either (Fig. 1). This phenomenon may lead to points from different classes mutually overlapping, and thus result in unsatisfactory classification results. We prefer the distributions of the data within each class to be tight such that the features are more discriminative. Detailed introduction will be presented in Section III.

To overcome the above side effect for cross-domain classification brought about by MMD, we believe that the intrinsic

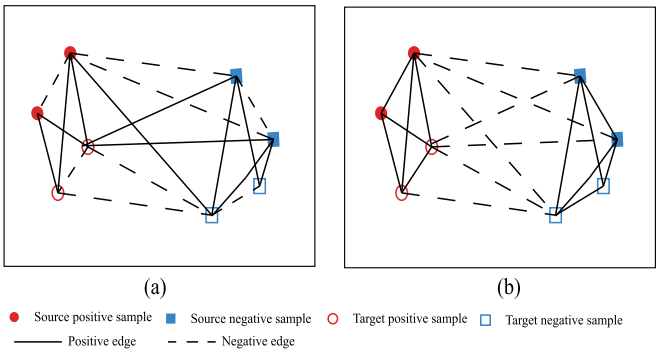


Fig. 1. (a) Intrinsic graph of empirical MMD minimization. Vertices from different domains are friends, while those from the same domain are enemies. Labels do not play a role in weight assignment. (b) The expected graph in which weights are assigned exactly consistently with labels. In both figures, only part of the edges are drawn for clarity.

graph of the MMD-based domain adaptation algorithms needs to be revised. The revision should lead to the new similarity matrix assigning positive weights to edges connecting same-label vertices.

In this paper, we propose a graph embedding framework in which i) a complementary term is appended to the empirical MMD to revise the similarity matrix of the intrinsic graph, and ii) constraints from penalty graphs can be designed. With the new intrinsic graph, the learned feature representations are expected to be more discriminative in both source and target domains than the original. Furthermore, constraints from various penalty graphs are involved such that certain unwanted geometric property can be avoided or suppressed. Within the framework, domain adaptation methods regarding MMD can be developed by merely designing two graphs.

Using the framework, we propose two strategies for intrinsic graph revision. Based on the two strategies, we design four kinds of penalty graphs inspired by classic dimensionality reduction or metric learning methods. In total, four models are exploited within the framework.

In summary, our contributions are three folds.

- We study the property of MMD from the point of view of graph embedding, and prove that in cross-domain classification tasks, the MMD metric can negatively affect the discriminative property of the samples, and that merely reducing the source and target distribution distance is insufficient for cross-domain classification. As far as we know, such study has not been done before.
- Based on the analysis, a graph embedding framework for MMD-based domain adaptation algorithms is proposed. In the framework, the intrinsic graph is revised to achieve better within-class compactness, and the constraints from penalty graphs are designed to avoid certain similarity property of data.
- We introduce two sorts of intrinsic graph revision strategies and exploit four different existing models using the framework. The four models are developed with specific penalty graphs in the constraint. The models inspired from dimensionality reduction methods are aimed to deal with the negative effect of MMD, and unified in our graph embedding framework. Comprehensive experiments conducted on cross-domain visual benchmark datasets show that our models outperform other state-of-the-art methods.

The remainder of the paper is structured as follows. Some previous work related to our study will be briefly reviewed in the next section. Introduction of graph embedding and a detailed analysis on the MMD are presented in Section III. In Section IV, we give our proposed graph embedding framework. We then utilize the framework to revise the MMD intrinsic graph with two strategies, and further propose four specific models in Section V. We experimentally evaluate the proposed models on a series of cross-domain datasets in Section VI. Finally, we conclude our research in Section VII.

II. Related Work

A. Domain Adaptation

There has been fruitful research on domain adaptation. Many good review papers on this issue are available [36], [39]. The research regarding MMD is the most relevant to our research.

Maximum Mean Discrepancy Embedding (MMDE) [34] introduces the MMD into the Maximum Variance Unfolding (MVU) technique and proposes a dimensionality reduction method for transfer learning. Transfer Component Analysis (TCA) [35] directly minimizes the MMD distance between distributions, together with a Frobenius norm regularizer and a variance preservation constraint. Joint Distribution Adaptation (JDA) [29] utilizes the predicted labels of target samples and reduces the divergences of both marginal and conditional distributions. The label and structural consistency of the unlabeled target data is further exploited in [22].

The approaches mentioned above concentrate on reducing distribution divergence between source and target domains through the MMD term, but fail to take into consideration the discriminative knowledge of the sample points which can boost the classification accuracy.

The most structurally relevant research is proposed in [1], [16], [17], [27]. Domain adaptation metric learning (DAML) [16] focuses on metric adaptation and integrates MMD into a conventional metric learning algorithm. Domain Invariant Projection (DIP) [1] learns a projection to a low-dimensional latent space, and further exploits the source label information for class clustering. The Scatter Component Analysis (SCA) [17] gives a framework for both domain adaptation and domain generalization which quantifies each term, including MMD, through scatter. The *within-class scatter* is also used in the algorithm. Domain invariant and class discriminative (DICD) representation learning [27] considers both intra-class similarity and inter-class dissimilarity. SCA and DICD have very similar forms. While these algorithms are indeed superior to simple MMD-based ones, a mathematical analysis has not been given. We present a theoretical explanation on why reducing within-class scatter works in the algorithms, and provide a unified framework for these methods. Furthermore, we show that various geometric property information can be leveraged besides sample scatter by designing different constraints.

Another widely used distribution metric is the Wasserstein distance [5], [40]. Wasserstein distance has a similar form to MMD, but is independently defined, and is minimized with different underlying strategies. To be specific, the minimal Wasserstein distance usually needs to be searched by gradient descent, while in classic MMD methods such as TCA, JDA and SCA, the optimal solution can be solved analytically.

B. Graph Embedding

Graph embedding is a technique for exploiting data structure and network information. So far, many graph embedding methods have been proposed [19]. The category of methods that is most related to our research is the factorization based methods. These methods represent the connections between graph vertices with a matrix. Famous algorithms include Locally Linear Embedding (LLE) [37], Laplacian Eigenmap [2], Cauchy Graph Embedding [31], etc. Laplacian Eigenmap aims to preserve the distances between the embeddings of vertices according to the weights in a Laplacian matrix, with a scale normalization constraint. In our paper, we study MMD using a similar idea to Laplacian Eigenmap, but replace the constraint with one related to another graph embedding problem.

A lot of research about Laplacian matrix based graph embedding formulations has been made [4], [44]–[46]. Brand [4] integrated nonlinear dimensionality reduction into graph embedding with side information about the vertices. Laplacian Eigenmap can be reformulated within this work. Yan *et al.* [44] proposed a graph embedding framework for dimensionality reduction including an intrinsic and a penalty graph. This framework is more general. Yang *et al.* [45] introduced a non-negative graph embedding formulation applicable for supervised and semi-supervised problem, which is powerful at classification. The formulation in Yang's work also involves two graphs, one for the favorite relationship and one for the unfavorable. Zhang *et al.* [46] further presented a robust non-negative graph embedding framework capable of dealing with noisy data, unreliable graphs, and noisy labels.

Conventional graph embedding algorithms are based on the underlying assumption that all the data can be treated as one sample set. However, in domain adaptation problems, data can be partitioned into different domains. And the domain related properties of MMD based models can be analyzed using a graph.

III. Analysis on MMD in Domain Adaptation

In this section, we first give a brief description on MMD in subsection A, including notations in the paper, the definition of MMD, as well as its general usage in domain adaptation. Then we introduce graph embedding and the concept of Laplacian matrix in subsection B. The basic form of graph embedding algorithms is also given. Finally, we present our analysis on MMD minimization from the point of graph embedding to study how MMD distance can be improved.

A. Introduction of MMD

1) Notations: We denote by $X_S = \{x_i\}_{i=1}^{N_S} \subset \mathbb{R}^d$ the source-domain data, and by $Y_S = \{y_i\}_{i=1}^{N_S} \subset \{1, 2, \ldots, C\}$ the corresponding source labels. Similarly, we denote by $X_T = \{x_j\}_{j=N_S+1}^{N} \subset \mathbb{R}^d$ the target-domain data, but with no labels available. $N = N_S + N_T$ is the number of total sample points. We assume that X_S is sampled from distribution Q_S, and X_T is from distribution Q_T which is different, i.e., $Q_S \neq Q_T$. Therefore, our goal is to discover a new feature representation that is common to both domains, based on which we learn a classifier from the labeled source data adaptive to the target data.

2) Definition: The maximum mean discrepancy was initially introduced as a test statistic for the hypothesis testing on whether two sets of observations are generated by the same distribution [3]. Let p, q be two independent probability distributions, and $\mathbb{E}_{x\sim p}[f(x)]$ denotes the mathematical expectation of $f(x)$ with x under probability density p. The premier definition of MMD between p and q is

$$\mathrm{MMD}(\mathcal{F}, p, q) \triangleq \sup_{f\in\mathcal{F}} \|\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]\|, \quad (1)$$

where $\mathcal{F}$ is a class of functions. It has been proved that when $\mathcal{F}$ is universal, then $\mathrm{MMD}(\mathcal{F}, p, q) = 0$ if and only if $p = q$. Typically, $\mathcal{F}$ will be chosen in a reproducing kernel Hilbert space (RKHS). Without loss of generality, $\mathcal{F}$ is set to be the unit ball in the RKHS.

In practice, we always use the square MMD in order for calculation with kernel functions:

$$\begin{aligned}
&\mathrm{MMD}^2(\mathcal{F}, p, q)\\
&\quad= \left(\sup_{\|f\|_{\mathcal{H}}\leqslant 1} \|\mathbb{E}_p[f(x)] - \mathbb{E}_q[f(y)]\|\right)^2\\
&\quad= \|\mathbb{E}_p[\phi(x)] - \mathbb{E}_q[\phi(y)]\|_{\mathcal{H}}^2\\
&\quad= \mathbb{E}_{p,p}[k(x, x')] + \mathbb{E}_{q,q}[k(y, y')] - 2\mathbb{E}_{p,q}[k(x, y)]. \quad (2)
\end{aligned}$$

The map $\phi(\cdot)$ is the *feature space map* from x to the RKHS $\mathcal{H}$, $\|\cdot\|_{\mathcal{H}}$ indicates the norm in space $\mathcal{H}$, and $k(x, x') = \langle\phi(x), \phi(x')\rangle_{\mathcal{H}}$ is the reproducing kernel of $\mathcal{H}$. A detailed proof of these equations can be found in [20]

In most popular domain adaptation algorithms regarding the MMD, the square of an empirical estimate, together with a regularizer, is usually minimized. The basic optimization problem can be expressed as

$$\min_{\varphi\in\mathcal{A}} \mathcal{J}(\varphi) = \left\| \frac{1}{N_S}\sum_{i=1}^{N_S} \varphi(x_i) - \frac{1}{N_T}\sum_{j=N_S+1}^{N} \varphi(x_j) \right\|^2 + \lambda \|\varphi\|_{\mathcal{H}}^2 . \quad (3)$$

$\mathcal{A} \subseteq \mathcal{H}$ is some function class defined by certain constraints, φ either is a m-dimensional vector-valued function, or represents the feature map corresponding to certain kernel function, i.e. $m = \infty$. λ is the trade-off coefficient. Some approaches [22], [27], [29] further utilize the label information and minimize both the marginal and conditional distribution distances.

The corresponding optimization problem is

$$\min_{\varphi\in\mathcal{A}} \mathcal{J}(\varphi) + \sum_{c=1}^{C} \left\| \frac{1}{n_S^c} \sum_{\substack{i\leqslant N_S \\ y_i=c}} \varphi(x_i) - \frac{1}{n_T^c} \sum_{\substack{j\geqslant N_S+1 \\ y_j=c}} \varphi(x_j) \right\|^2, \tag{4}$$

where C is the number of classes, and n_S^c, n_T^c are number of points belonging to class c in source and target domains, respectively. Note that the labels of target points are unavailable in model training. Practically we use pseudo labels $\{\hat{y}_j\}$ predicted by some simple classifier trained from source data as replacement, and update the pseudo labels iteratively. The pseudo labels will approach the ground truth with iterative updating according to [29]. For clarity, we temporarily denote the target labels as the true labels $\{y_j\}$ in our analysis, instead of $\{\hat{y}_j\}$.

B. Introduction of Graph Embedding

Let $G = \{V,\ W\}$ be an undirected signed weighted graph with vertex set $V = \{v_1, v_2, \ldots, v_N\}$ and similarity matrix $W \in \mathbb{R}^{N\times N}$ which is real symmetric. Each element W_{ij} assigns the edge between vertex v_i and v_j a weight which measures the similarity. A negative edge connects two dissimilar vertices (enemies), and a positive one connects similar vertices (friends). The magnitude of the weight represents the degree of similarity. The graph embedding of a graph G is an algorithm to learn feature representations for the vertices that best characterize the similarity relationship according to W. The Laplacian matrix L of a graph G is defined as

$$\begin{aligned} &L = D - W, \\ &D \text{ is diagonal}, \quad D_{ii} = \sum_{j\neq i} W_{ij}, \end{aligned}$$

C. Basic Form of Graph Embedding Algorithms

In a graph embedding algorithm, two graphs are involved: the *intrinsic graph* $G = \{V,\ W\}$ and the *penalty graph* $G^p = \{V,\ W^p\}$. The algorithm requires that the learned feature representations $Z = \{z_1, z_2, \ldots, z_N\}$ preserve the similarity relationship of the intrinsic graph G, meaning that the distance between closer friends should be smaller, while enemies should have larger distances. The penalty graph G^p usually characterizes certain similarity property that is unfavorable and to be suppressed.

Take the 1-dimensional case for example, i.e., $Z = [z_1, z_2, \ldots, z_N]^{\mathrm{T}} \subset \mathbb{R}^{N\times 1}$, and we give the optimization problem of a graph embedding algorithm as follows:

$$z^* = \mathop{\arg\min}_{\sum_{i\neq j}\|z_i - z_j\|^2\ W_{ij}^p = 1} \sum_{i\neq j} \|z_i - z_j\|^2\ W_{ij} = \mathop{\arg\min}_{Z^{\mathrm{T}}PZ=1} Z^{\mathrm{T}}LZ, \tag{5}$$

where P is the Laplacian matrix for penalty graph G^p. The graph embedding constraint sets the sum of square distances weighted by W^p to be a constant 1 and causes vertices in the learned subspace to avoid the similarity relationship characterized by G^p. This constraint $Z^{\mathrm{T}}PZ = 1$ is acceptable because the constraint and the object function of (5), $Z^{\mathrm{T}}LZ$, are homogeneous, and changing the constant 1 to any positive value c^2 merely makes the optimal solution z^* become cz^*, which does not lead to any essential difference.

Linearization: When the feature representation is learned through a linear projection, i.e., $Z = X^{\mathrm{T}}a$, where a is the projection vector, the optimization problem becomes

$$z^* = \mathop{\arg\min}_{a^{\mathrm{T}}XPX^{\mathrm{T}}a=1} a^{\mathrm{T}}XLX^{\mathrm{T}}a = \mathop{\arg\min}_{a} \frac{a^{\mathrm{T}}XLX^{\mathrm{T}}a}{a^{\mathrm{T}}XPX^{\mathrm{T}}a}. \tag{6}$$

The second equation is because of the quadratic property of the object function and the constraint.

Kernelization: For the nonlinear problems, the kernelization technique can be utilized. Consider the kernel function $k_{ij} = k(x_i, x_j)$, and kernel gram matrix $K = [k_{ij}]_{N\times N}$. By using the Representer theorem, we formulate the kernel version of the optimization problem as

$$z^* = \mathop{\arg\min}_{\alpha^{\mathrm{T}}KPK^{\mathrm{T}}\alpha=1} \alpha^{\mathrm{T}}KLK^{\mathrm{T}}\alpha = \mathop{\arg\min}_{\alpha} \frac{\alpha^{\mathrm{T}}KLK^{\mathrm{T}}\alpha}{\alpha^{\mathrm{T}}KPK^{\mathrm{T}}\alpha}. \tag{7}$$

D. Analysis on MMD Minimization as Graph Embedding

We now understand the empirical MMD from the point of view of *graph embedding*. We first rewrite the empirical MMD in a matrix form:

$$\begin{aligned} &\mathrm{MMD}^2(\varphi, X_S, X_T) \\ &\quad = \left\| \frac{1}{N_S} \sum_{i=1}^{N_S} \varphi(x_i) - \frac{1}{N_T} \sum_{j=N_S+1}^{N} \varphi(x_j) \right\|^2 \\ &\quad = \mathrm{Tr}[\varphi(X) M^0 \varphi(X)^{\mathrm{T}}], \end{aligned}$$

where $\varphi(X) = [\varphi(x_1), \varphi(x_2), \ldots, \varphi(x_N)] \in \mathbb{R}^{m\times N}$ is the data matrix. $\mathrm{Tr}[\cdot]$ is the trace of a square matrix. It is easy to calculate the matrix M^0 as

$$M^0 = \begin{bmatrix} \frac{1}{N_S^2}\mathbf{1}_{N_S\times N_S} & -\frac{1}{N_S N_T}\mathbf{1}_{N_S\times N_T} \\ -\frac{1}{N_S N_T}\mathbf{1}_{N_T\times N_S} & \frac{1}{N_T^2}\mathbf{1}_{N_T\times N_T} \end{bmatrix}.$$

Here we denote $\mathbf{1}$ as the matrix of all elements being 1. M^0 is actually a matrix that can be defined as

$$\begin{aligned} &M^0 = D^0 - W^0, \\ &D^0 \text{ is diagonal}, \quad D_{ii}^0 = \sum_{j\neq i} W_{ij}^0, \end{aligned}$$

$$W_{ij}^0 = \begin{cases} -\frac{1}{N_S^2} & \text{if } i, j \leqslant N_S \text{ and } i \neq j \\ -\frac{1}{N_T^2} & \text{if } i, j > N_S \text{ and } i \neq j \\ \frac{1}{N_S N_T} & \text{if } i \leqslant N_S < j \text{ or } j \leqslant N_S < i \\ 0 & \text{if } i = j. \end{cases}$$

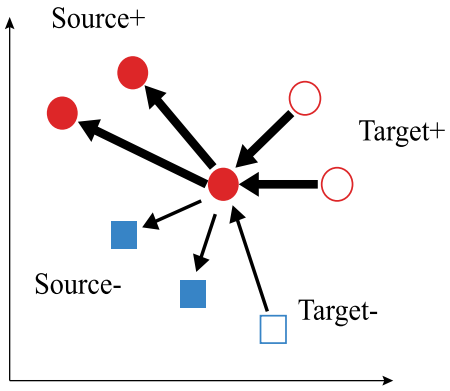


Fig. 2. The adjacency relationship of the intrinsic graph of marginal and conditional MMD. For clarity, only the edges of one source positive sample are shown. The arrows point in the directions for MMD minimization, i.e., out for exclusion and in for attraction, and the arrow thickness roughly indicates the magnitude of edge weights.

Similarly, we can get the conditional counterparts of matrices M^0 and W^0 denoted by M^c and W^c, respectively. The similarity matrix W^c is

$$W_{ij}^c = \begin{cases} -\dfrac{1}{(n_S^c)^2} & \text{if } i, j \leqslant N_S,\ i \neq j \text{ and } y_i = y_j = c \\ -\dfrac{1}{(n_T^c)^2} & \text{if } i, j > N_S,\ i \neq j \text{ and } y_i = y_j = c \\ \dfrac{1}{n_S^c n_T^c} & \text{if } i \leqslant N_S < j \text{ or } j \leqslant N_S < i,\ y_i = y_j = c \\ 0 & \text{if otherwise.} \end{cases}$$

As a result, the MMD minimization in optimization problem (4) equals to finding the feature representations that best characterize the similarity relationship between pairs of vertices in $G = \{X, W^0 + \sum_c W^c\}$. It is clear that the graph G connects all pairs of data points within the same domain and class with *negative* weights, indicating that the points of the same class are deemed as dissimilar *enemies*. Therefore, the distance between each pair of same-label points is enlarged to minimize the object function, resulting in the *within-class compactness* reduction, as illustrated in Fig. 2. However, in classification tasks, the points of the same class are supposed to be clustered tighter for higher accuracy. The optimization of problem (4) is quite on the contrary of this goal.

Relation to Wasserstein Distance: Minimizing the Wasserstein distance between two domains can also achieve domain adaptation [5], [40]. The minimization of Wasserstein distance has the form of graph embedding, but there exists a major difference between intrinsic graphs of the two metrics: the weights in MMD graph are pre-defined, while those in Wasserstein graph are not fixed but need to be searched. In this paper, the representation learning based on a fixed graph is discussed.

IV. Framework for MMD-Based Domain Adaptation Algorithms

In this section, we propose our graph embedding framework for MMD-based domain adaptation algorithms. Specifically, we present the linear form of the framework, along with its optimization procedure.

A. The Graph Embedding Framework

The analysis on MMD suggests that, the MMD term is helpful in reducing distribution distance, but ignores within-class compactness. To make up for the defect, a revision is necessary for the MMD-based domain adaptation algorithms. We aim to propose a revised object function for the MMD-based domain adaptation algorithms which can overcome the drawback while preserving the advantage of MMD.

Since we see the MMD minimization as graph embedding of the graph $G = \{X, W^0 + \sum_c W^c\}$, the revision is supposed to adjust the edge weights of G such that the similarity measurement is consistent with the labels. The direct way is to add a revision graph $\Delta G = \{X, \Delta W\}$ to the original graph such that the new intrinsic graph becomes $G + \Delta G$. Let

$$R = D^r - \Delta W,$$
$$D^r \text{ is diagonal}, \quad D_{ii}^r = \sum_{j \neq i} \Delta W_{ij}.$$

Then the optimization problem of the revised MMD-based feature representation learning algorithms can be written in a unified formulation:

$$\min_{\operatorname{diag}(\varphi(X)^{\mathrm{T}} P \varphi(X)) = \mathbf{1}} \operatorname{Tr}\left[\varphi(X)\left(\sum_{c=0}^{C} M^c + R\right)\varphi(X)^{\mathrm{T}}\right] + \lambda \|\varphi\|_{\mathcal{H}}^2. \tag{8}$$

$\operatorname{diag}(\cdot)$ is the vector composed of the diagonal elements of a matrix. P is the Laplacian matrix for the penalty graph G^p. Remind that the constraint from the penalty graph is designed to avoid trivial solution and typically characterizes certain similarity property to be geometrically avoided or suppressed in the learned subspace. Specifically, the penalty graph can be designed to characterize the distance between different classes such that the inter-class similarity is reduced during optimization. The latter three models proposed in our following section adopt penalty graphs in this strategy. They will be described in detail in Section V.

Based on the analysis in Section III-D, it can be stated that the revision by the graph ΔG is not merely an auxiliary to the MMD graph G, but a necessary complement. They can be treated as one term and always accompany each other in classifier training algorithms.

It is worth noting that, although the final optimal MMD distance may be affected due to the revising of the object function, a proper revision graph ΔG is not contradictory to domain discrepancy reduction. It is possible as long as the positive cross-domain edges in graph G are kept unchanged or enhanced, while the overall revised graph is improved only for the benefit of classification rather than domain alignment.

The framework can be utilized to develop different MMD-based domain adaptation algorithms to improve the performance of methods merely minimizing MMD distance. To design an algorithm within the framework, we only need to input the similarity matrices of the revision graph ΔG and penalty graph G^p.

Algorithm 1 Linear Graph Embedding Framework for MMD-Based Domain Adaptation Algorithms

Input: Labeled source samples $\{x_i, y_i\}_{i=1}^{N_S}$; unlabeled target samples $\{x_j\}_{j=N_S+1}^{N}$; coefficient λ; subspace dimensionality m; iteration number T.

1: Construct MMD matrix M, revision graph $\Delta G = \{X, \Delta W\}$ and penalty graph $G^p = \{X, W^p\}$, and set $M_c = \mathbf{0}, c = 1, 2, \ldots, C$.
2: **for** $t = 1 : T$ **do**
3: Calculate R and P, the Laplacian matrices of ΔG and G^p, respectively.
4: Solve the generalized eigencomposition problem in (11) and construct the projection matrix $A \in \mathbb{R}^{d\times m}$.
5: Train an adaptive classifier F on the reconstructed source samples $\{A^{\mathrm{T}}x_i, y_i\}_{i=1}^{N_S}$.
6: Update the target pseudo labels $\{\hat{y}_j\}_{j=N_S+1}^{N}$ and conditional matrices $\{M_c\}_{c=1}^{C}$ using F.
7: **end for**

Output: The final projection matrix A and the final adaptive classifier F.

B. Linear Form

Let the function φ be a simple linear projection, i.e., $\varphi(x) = A^{\mathrm{T}}x$, and define $\|\varphi\|_{\mathcal{H}}$ as the Frobenius norm $\|A\|_F = \sqrt{\mathrm{Tr}[A^{\mathrm{T}}A]}$. Then we have the linear version of the graph embedding framework as

$$\min_{\substack{a_u^{\mathrm{T}}XPX^{\mathrm{T}}a_u=1,\\ u=1,2,\ldots,m}} \mathrm{Tr}\left[A^{\mathrm{T}}\left(X(\sum_{c=0}^{C}M^c + R)X^{\mathrm{T}} + \lambda I\right)A\right], \quad (9)$$

where a_u is the u-th column of projection matrix A, and I is the identity matrix.

Further, we can yield the kernel graph embedding formulation by replacing the data matrix X by the gram matrix K.

C. Optimization

Algorithms within the linear graph embedding framework can be optimized in an analytical way. We derive the Lagrange function for problem (9) as

$$\mathcal{L}(A, \Theta) = \mathrm{Tr}\left[A^{\mathrm{T}}\left(X(\sum_{c=0}^{C}M^c + R)X^{\mathrm{T}} + \lambda I\right)A\right] + \mathrm{Tr}[(I - A^{\mathrm{T}}XPX^{\mathrm{T}}A)\Theta], \quad (10)$$

where $\Theta = \mathrm{diag}(\theta_1, \ldots, \theta_p)$ is a diagonal matrix, $\theta_1, \ldots, \theta_m$ are Lagrange multipliers. Set $\dfrac{\partial\mathcal{L}(A,\Theta)}{\partial A} = 0$ and obtain generalized eigendecomposition

$$\left(X(\sum_{c=0}^{C}M^c + R)X^{\mathrm{T}} + \lambda I\right)A = XPX^{\mathrm{T}}A\Theta. \quad (11)$$

Therefore, the optimization of problem (10) is reduced to finding the eigenvectors corresponding to the p smallest eigenvalues in (11).

The complete procedure is summarized in Algorithm 1.

V. Algorithms Developed From Framework

The graph embedding framework can be adopted to develop domain adaptation algorithms by designing two graphs, ΔG and G^p. We present two revision strategies and four graph embedding models based on the framework. Each model is inspired by one classic dimensionality reduction or metric learning method which can be reformulated as a graph embedding algorithm. To simplify the discussion, only the linear version of the framework is considered in this section. The kernel version has the identical form except for the gram matrix K in place of X.

A. Strategies for Intrinsic Graph Revision

1) Global Compactness: The key point in the analysis on MMD is that the within-domain/class edge weights are negative in the MMD intrinsic graph. Therefore, the proposed idea is simple: to add weights to the MMD graph. We choose not to add the within-domain scatter term because it is to reduce the compactness within each class that is key to classification.

We design the similarity matrix ΔW of the revision graph ΔG as $\Delta W = \mathrm{diag}\,(\Delta W_S, \Delta W_T)$, where

$$(\Delta W_S)_{ij} = \begin{cases}\alpha_c & \text{if } i \neq j \text{ and } y_i = y_j = c\\ 0 & \text{otherwise;}\end{cases} \quad (12)$$

$$(\Delta W_T)_{ij} = \begin{cases}\beta_c & \text{if } i \neq j \text{ and } \hat{y}_i = \hat{y}_j = c^1\\ 0 & \text{otherwise.,}\end{cases} \quad (13)$$

α_c and β_c are non-negative weights of edges connecting source and target vertices of class c, respectively. $\hat{y}_i$ is the pseudo label of x_i in target domain. Further, we have $R = \mathrm{diag}\,(R_S, R_T)$, where

$$(R_S)_{ij} = \begin{cases}(n_S^c - 1)\alpha_c & \text{if } i = j \text{ and } y_i = c\\ -\alpha_c & \text{if } i \neq j \text{ and } y_i = y_j = c\\ 0 & \text{otherwise;}\end{cases}$$

$$(R_T)_{ij} = \begin{cases}(n_T^c - 1)\beta_c & \text{if } i = j \text{ and } \hat{y}_i = c\\ -\beta_c & \text{if } i \neq j \text{ and } \hat{y}_i = \hat{y}_j = c\\ 0 & \text{otherwise.}\end{cases}$$

Since the revision graph assigns all the edges within each class with non-negative weights, we refer to the strategy as *global compactness*.

Remark: The edge weights α_c and β_c are left unfixed. How the weights are balanced should depend on the samples. For example, for well separated sample set, the weights need not to be large, while for class-imbalanced dataset, the weights for different classes should be appropriately varied from each other. Relevant experiments are presented in Section VI.

2) Local Neighborhood: The effectiveness of global compactness is based on one assumption that the data within each class is of a unimodal distribution, for example, Gaussian distribution. While dealing with distributions with multimodal

[1]Technically, the subscripts should be $N_S + i$ and $N_S + j$ according to the notations in Section III-A-1. But for clarity of writing, we will still use i and j for target data without confusion in the rest of the paper.

support, however, we prefer concentrating on local neighbors of each point. Therefore, the *local neighborhood* strategy is proposed.

We define the k *same-class neighbors* of a sample x_i as k other points with the same label y_i that are the closest to x_i. We let $\eta_{ij} = 1$ when x_j is a same-class neighbor of x_i within the same domain, and $\eta_{ij} = 0$ otherwise. Then we have for any x_i (from either source or target domain) labeled by class c,

$$(\Delta W_S)_{ij} = (\eta_{ij} + \eta_{ji})\alpha_c,$$
$$(\Delta W_T)_{ij} = (\eta_{ij} + \eta_{ji})\beta_c.$$

This definition means that the local neighborhood graph only enhances similarity weights between each point and its same-class neighbors. The Laplacian matrix of local neighborhood graph is written as $R = \mathrm{diag}(R_S, R_T)$, where

$$(R_S)_{ij} = \begin{cases} (k + k^i)\alpha_c & \text{if } i = j \\ -(\eta_{ij} + \eta_{ji})\alpha_c & \text{if } i \neq j; \end{cases}$$
$$(R_T)_{ij} = \begin{cases} (k + k^i)\beta_c & \text{if } i = j \\ -(\eta_{ij} + \eta_{ji})\beta_c & \text{if } i \neq j. \end{cases}$$

Here k is the selected number of same-class neighbors, and k^i is the number of points that have x_i as one of its same-class neighbors.

B. Models

We present four models using the Graph Embedding Framework (abbreviated to GEF) based on classic dimensionality reduction or metric learning methods including PCA, LDA, LMNN and MFA. In the rest of the paper the models are denoted as "GEF-PCA", "GEF-LDA", "GEF-LMNN", "GEF-MFA", respectively. These models have different motivations, but share the same formulation. Specifically, in the last three models, the penalty graph are designed to constrain the relationship between points of different classes, which is supplementary to the revision graphs of the previous subsection.

1) Principal Component Analysis Based Model: Principal Component Analysis (PCA) [25] is a well-known dimensionality reduction algorithm focusing on information preservation. PCA is also quite popular in domain adaptation algorithms [29], [35]. Usually, it is used as a sample variance preservation constraint to avoid trivial solutions. The variance of the u-th projected feature is $a_u^{\mathrm{T}} X H X^{\mathrm{T}} a_u$, where $H = I - (1/N)\mathbf{1}\mathbf{1}^{\mathrm{T}}$ is the centering matrix. So we have the optimization problem as

$$\min_A \ \mathrm{Tr}\left[A^{\mathrm{T}}\left(X(\sum_{c=0}^{C} M^c + R)X^{\mathrm{T}} + \lambda I\right)A\right]$$
$$\text{s.t. } a_u^{\mathrm{T}} X H X^{\mathrm{T}} a_u = 1, u = 1, 2, \ldots, m.$$

It is obvious that the centering matrix H is a Laplacian matrix with the similarity matrix $W^p = (1/N)\mathbf{1}\mathbf{1}^{\mathrm{T}}$. The penalty graph of PCA constraint connects all pairs of vertices with equal weights $1/N$, manifesting that PCA constraint suppresses the sample structures in which all the points are tightly gathered such that neither distribution nor class information is preserved, one extreme case of which is that the solution is trivial, i.e., all the data are projected to the origin.

In GEF-PCA we choose the global compactness term since PCA extracts information from the whole sample set. Note that when the coefficients $\alpha_c, \beta_c = 0$, i.e., $R = 0$, the model reduces to Joint Distribution Adaptation (JDA). But the term regarding within-class compactness is necessary, as we discussed in Section III.

It is worth noting that, GEF-PCA is exactly the same as DICD-S proposed in [27]. However, it can be seen that DICD-S is one instance of the models derived from a general framework, and there exist different object functions and constraints. Three other kinds of models will be presented in the following.

2) Linear Discriminant Analysis Based Model: Linear Discriminant Analysis (LDA) [14], [33] searches for the features that best discriminate among classes, rather than preserve information, in the underlying space. Formally, LDA yields the largest differences between classes by minimizing within-class scatter and maximizing inter-class scatter. The global compactness graph characterize exactly the within-class scatter in the revised object function. The inter-class scatter $\sum_{y_i \neq y_j} \|a_u^{\mathrm{T}} x_i - a_u^{\mathrm{T}} x_j\|^2$ is constrained to be a fixed number for each dimensionality. We implement LDA in source and target domains respectively. It follows the linear graph embedding formulation as (9). We define $y_{ij} = 1$ when x_i and x_j share the same label, and $y_{ij} = 0$ if not. In the penalty similarity matrix $W^p = \mathrm{diag}(W_S^p, W_T^p)$, we have $(W_{S/T}^p)_{ij} = 1 - y_{ij}$. Correspondingly, the Laplacian matrix $P_{LDA} = \mathrm{diag}(P_S, P_T)$, where

$$(P_S)_{ij} = \begin{cases} N_S - n_S^c & \text{if } i = j \text{ and } y_i = c \\ -1 & \text{if } i \neq j \text{ and } y_i \neq y_j \\ 0 & \text{otherwise;} \end{cases}$$
$$(P_T)_{ij} = \begin{cases} N_T - n_T^c & \text{if } i = j \text{ and } \hat{y}_i = c \\ -1 & \text{if } i \neq j \text{ and } \hat{y}_i \neq \hat{y}_j \\ 0 & \text{otherwise.} \end{cases}$$

GEF-LDA differs from GEF-PCA in that the former is designed for classification tasks while the latter aims at domain alignment without significant initial information loss.

3) Large Margin Nearest Neighbor Based Model: Large Margin Nearest Neighbor (LMNN) [42] is a distance metric learning algorithm. Its goal is to learn a metric that separates samples from different classes by a large margin and ensures the k-nearest neighbors of each sample are from the same class.

The cost function of LMNN has two terms. The first one is to penalize large distances between each sample and its same-class neighbors, and the second one penalizes the *impostors* that invade the perimeters established by each sample and its same-class neighbors that differently labeled points should not enter. We again use $\eta_{ij} \in \{0, 1\}$ to denote whether x_j is a same-class neighbor of x_i, and $y_{ij} \in \{0, 1\}$ to denote whether x_i and x_j share the same label ($y_{ij} = 1$) or not ($y_{ij} = 0$).

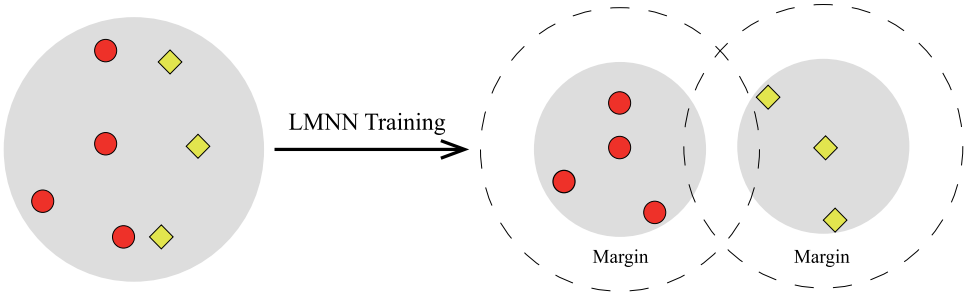


Fig. 3. An illustration for the training process of LMNN.

Then we have the cost function as

$$\varepsilon(A) = \sum_{ij} \eta_{ij} \|A^{\mathrm{T}}(x_i - x_j)\|^2 + \rho \sum_{ijl} \eta_{ij}(1 - y_{il})[1 + \|A^{\mathrm{T}}(x_i - x_j)\|^2 - \|A^{\mathrm{T}}(x_i - x_l)\|^2]_+,$$

where $[z]_+ = \max(0, z)$ is the standard hinge loss, and ρ is a positive constant. A schematic illustration of the learning process induced by the cost function is presented in Fig. 3.

We would like to derive a model inspired from the LMNN within our framework. The first term of LMNN cost function is exactly the proposed local neighborhood term. In order to integrate the LMNN model into the framework, we transform the second term into an equality constraint. We notice that the hinge loss only penalizes small inter-class distances, but does not encourage them to be significantly larger than distances from the sample to same-class neighbors. This suggests that we may fix the margin width to form the equality constraint as

$$\sum_{\substack{\eta_{ij}=1,\\ y_{il}=0}} [\|a_u^{\mathrm{T}}(x_i - x_l)\|^2 - \|a_u^{\mathrm{T}}(x_i - x_j)\|^2] = 1, u = 1, 2, \ldots, m.$$

The similarity matrix is

$$(W_{S/T}^p)_{ij} = -(\eta_{ij} + \eta_{ji})(N_{S/T} - n_{S/T}^c) + 2(1 - y_{ij})k,$$

for any x_i such that $y_i = c$ or $\hat{y}_i = c$, and $W^p = \mathrm{diag}(W_S^p, W_T^p)$. k is the number of same-class neighbors selected. We use the subscript S/T to claim that this matrix is defined in source and target domains, respectively.

It can be seen that the similarity weights are decided by the size of each class, and the number of neighbors involved. Two observations about GEF-LMNN can be made: i) a small-size class is supposed to have tighter inner connection for better performance in nearest neighbor classifier; ii) more selected neighbors can lead to larger distances between points having different labels.

Compared to GEF-LDA model, GEF-LMNN model concentrates on local neighborhoods instead of global structure. As illustrated in [42], this method particularly benefits the k-nearest neighbor classification accuracy for classes with multimodal support.

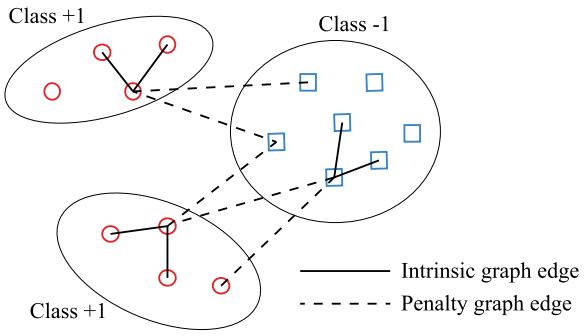


Fig. 4. The adjacency relationship of the intrinsic and penalty graphs for MFA with $k_1 = k_2 = 2$. For clarity, only the edges of three points (two from class +1 and one from class -1) are shown.

4) Marginal Fisher Analysis Based Model: Marginal Fisher Analysis (MFA) is a dimensionality reduction algorithm designed using a unified graph embedding framework proposed in [44]. MFA pulls each sample nearer to its k_1 same-class neighbors and penalizes small distances between each sample and its k_2 nearest neighbors with different labels. The adjacency relationship is shown in Fig. 4. The algorithm is similar to LDA, but is more general for discriminant analysis in that local information is leveraged without need of prior assumption on data distributions, while LDA only deals with data with each class following normal distributions.

To integrate MFA into our framework, it is obvious that the local neighborhood strategy should be adopted in the object function. The constraint matrix for GEF-MFA $P_{MFA} = \mathrm{diag}(P_S, P_T)$ is still a block diagonal matrix, each block being a Laplacian matrix for the penalty graph of the corresponding domain. The penalty weight matrix can be derived from the graph as illustrated in Fig. 4. Take the source domain block for example. We have $(W_S^p)_{ij} = \zeta_{ij} + \zeta_{ji}$, where $\zeta_{ij} = 1$ when x_j is one of the k_2 nearest neighbors of x_i with different labels, and $\zeta_{ij} = 0$ otherwise. The calculation of W_T^p is the same.

VI. Experiments

To evaluate the instantiated models within the framework, we have performed experiments on three real world vision cross-domain datasets. We will begin by presenting the benchmarks and experimental setting. Then comparison results along with some property analysis are presented.

TABLE I
PROPERTY OF THE BENCHMARK DATASETS

Dataset	Type	# Samples	# Features	# Classes
USPS	Digit	1800	256	10
MNIST	Digit	2000	256	10
AMAZON	Object	958	800(4096)	10
Webcam	Object	295	800(4096)	10
DSLR	Object	157	800(4096)	10
Caltech-256	Object	1123	800(4096)	10
CMU-PIE	Face	11554	1024	68

A. Datasets

We have conducted our experiments on five benchmark datasets: Office [18], [38], Caltech256 [21], [29], USPS [30], MNIST [30], and CMU-PIE [41]. A summary of the properties of these datasets is in Table I.

USPS is a hand-written digit dataset containing 7291 training images and 2007 testing images of size 16×16. The images belong to 10 classes (single digits 0-9). We sample 1800 images from USPS dataset as a domain in our digit recognition task. **MNIST** is another widely used standard hand-written digit dataset. MNIST consists of 60000 training images and 10000 testing images of size 28×28 and shares the same 10 classes as USPS. 2000 samples are randomly chosen to form a domain. The samples from MNIST are uniformly resized to 16×16 and vectorized. To obtain a common feature space shared by both domains, the grey levels of all images are normalized.

Office contains three real world object datasets. *Amazon* (A) images are photographs taken in studio environment downloaded from online merchants. *Webcam* (W) consists of images under natural lighting with low resolution. *DSLR* (D) images are taken by a high-resolution digital SLR camera. These three datasets have 31 categories. **Caltech256** (C) is a standard object database with 256 categories. 10 object categories included in all four domains are extracted. The datasets are preprocessed by two types of feature extraction methods. *SURF-BoW* features are extracted by SURF and quantized into 800-bin histograms. The final samples are 800 dimensional. *DeCAF*$_6$ features are the outputs of the neurons from the fully connected 6th layer of a deep convolutional neural network [9]. DeCAF$_6$ features form vectors with 4096 dimensions. Fig. 5 shows example images of these digit and object datasets.

CMU-PIE is a benchmark face database. The database has 32×32 images of 68 individuals with various pose, illumination and expressions. Five subsets are selected as five experimental domains, each corresponding to a distinct pose: PIE05 (left pose), PIE07 (upward pose), PIE09 (downward pose), PIE27 (frontal pose) and PIE29 (right pose). In this way, we can define 20 different adaptation tasks with domains following significantly different distributions. Fig. 6 shows example images of the CMU-PIE database.

The tasks are denoted in the "source $\rightarrow$ target" form, e.g., USPS $\rightarrow$ MNIST, C $\rightarrow$ A, PIE05 $\rightarrow$ PIE07.

B. Experimental Settings

We compare the four models based on the proposed framework with several state-of-the-art baseline methods. In all the sample transfer methods, we conduct the classification with a 1-Nearest Neighbor (1NN) classifier. The baseline methods for comparison are the following:

- *1NN* [6] is the baseline for all the experiments as a parameter-free classifier.
- *JDA*, Joint Distribution Adaptation [29], simply adopts MMD minimization procedure for domain sharing feature extraction.
- *TJM*, Transfer Joint Matching [30], aims to reduce the domain difference by jointly matching the features and reweighting the instances across domains
- *DTSL*, Discriminative Transfer Subspace Learning [43], preserves global and local structures of data by imposing joint low-rank and sparse constraints on the reconstruction coefficient matrix.
- *RTML*, Robust Transfer Metric Learning [8], builds a metric in a marginalized denoising fashion and low-rank constraint to enhance robustness to tackle noisy data.
- *SCA*, Scatter Component Analysis [17], finds a representation that trades between maximizing the separability of classes, minimizing the mismatch between domains, and maximizing the separability of data.
- *OT-GL*, Optimal Transport with Group-Lasso regularizer [5], perform the alignment of the representations in the source and target domains by a transportation matching plan with the class label information used.
- *DICD*, Domain Invariant and Class Discriminative feature learning [27], is an instance of our framework, and is similar to GEF-PCA, except that an intra-class scatter term is added to the object function.
- Ding et al. [7] proposes a semi-supervised deep domain adaptation architecture via coupled neural networks.

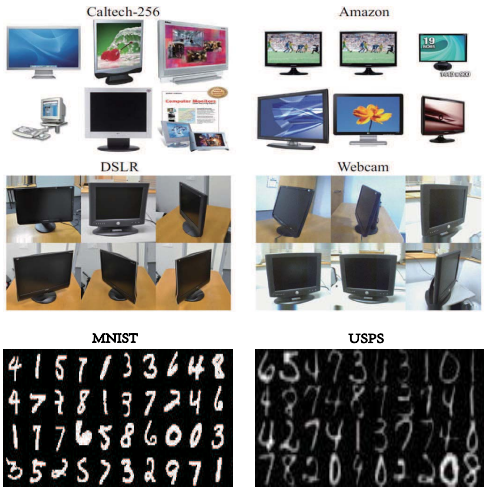


Fig. 5. Example images of Caltech-256, Office, MNIST, and USPS datasets.

Fig. 6. Example images of CMU-PIE database.

TABLE II
ACCURACY(%) ON HAND-WRITTEN DIGIT AND OBJECT DATASETS (SURF)

Tasks	1NN	JDA	TJM	RTML	SCA	OT-GL	GEF-PCA	GEF-LDA	GEF-LMNN	GEF-MFA
USPS→MNIST	44.70	59.90	54.90	61.82	48.00	50.50	60.90	60.25	**63.65**	60.95
MNIST→USPS	65.94	66.89	65.39	69.52	65.11	54.00	73.00	74.17	73.83	**76.22**
C→A	23.70	44.78	47.08	**49.26**	43.74	49.16	47.08	48.23	46.35	44.05
C→W	25.76	41.69	35.25	44.72	33.56	**48.47**	47.46	47.80	45.42	43.05
C→D	25.48	45.22	40.76	47.56	39.49	49.04	49.68	50.32	**52.87**	45.86
A→C	26.00	39.36	39.18	**43.68**	38.29	38.29	41.67	42.65	41.50	37.67
A→W	29.83	37.97	35.59	44.32	33.90	42.71	44.41	**46.44**	40.34	41.69
A→D	25.48	39.49	33.12	43.86	34.21	**51.59**	39.49	36.94	41.40	38.22
W→C	19.86	31.17	29.21	34.83	30.63	**37.31**	33.66	33.57	30.90	31.79
W→A	22.96	32.78	29.02	35.28	30.48	38.94	34.03	34.03	**41.34**	32.36
W→D	59.24	89.17	90.45	91.02	**92.36**	87.90	89.81	**92.36**	91.08	88.54
D→C	26.27	31.52	30.63	34.58	32.32	33.04	34.19	**35.44**	30.54	31.61
D→A	28.50	33.09	30.90	33.26	33.72	**36.85**	34.97	34.76	34.45	32.78
D→W	63.39	89.49	90.51	89.68	88.81	89.83	**90.85**	90.51	86.10	89.15
Average	34.79	48.75	46.57	51.67	46.04	50.55	51.51	**51.96**	51.41	49.57

TABLE III
ACCURACY(%) ON OBJECT DATASETS (DECAF6)

Tasks	1NN	JDA	TJM	RTML	SCA	OT-GL	Ding [7]	GEF-PCA	GEF-LDA	GEF-LMNN	GEF-MFA
C→A	85.70	89.77	89.77	90.62	89.46	82.99	86.12	**91.34**	91.23	89.46	90.50
C→W	66.10	83.73	78.64	85.38	85.42	73.90	86.24	88.81	**89.15**	88.81	86.10
C→D	74.52	86.62	85.99	89.32	87.90	78.98	80.13	**91.08**	88.54	85.35	90.45
A→C	70.35	82.28	79.43	**86.43**	78.81	77.11	74.89	83.97	83.62	83.17	84.86
A→W	57.29	78.64	75.93	80.26	75.93	69.83	67.98	78.64	76.27	81.02	**87.12**
A→D	64.97	80.25	82.17	84.36	85.35	71.97	81.93	**85.99**	82.17	78.98	80.25
W→C	60.37	83.53	75.78	83.13	74.80	71.86	72.49	83.88	83.97	**84.24**	82.19
W→A	62.53	90.19	86.12	**91.37**	86.12	77.56	79.43	89.25	89.04	90.61	90.61
W→D	98.73	**100.00**	**100.00**	**100.00**	**100.00**	92.36	99.46	**100.00**	**100.00**	**100.00**	**100.00**
D→C	52.09	85.13	78.90	85.72	78.09	77.03	66.34	**86.29**	**86.29**	85.04	85.66
D→A	62.73	91.44	89.46	91.86	89.98	84.45	74.98	**92.28**	92.17	91.44	91.02
D→W	89.15	98.98	98.98	98.98	98.64	94.58	96.57	98.98	98.98	99.32	**100.00**
Average	70.38	87.55	85.10	88.95	85.88	79.38	80.55	**89.21**	88.45	88.12	89.06

A semi-supervised class-wise adaptation manner is developed, and a multi-class classifier is simultaneously trained.

For fair comparison, we adopt consistent hyper-parameters for all the four proposed algorithms along with JDA and TJM, in that JDA and TJM are similar to our framework in object function and training procedure. The dimensionality of the learned subspace is 100 for all the tasks and algorithms. We set the regularizer parameter $\lambda = 0.01$ for the face recognition tasks (CMU-PIE dataset), and $\lambda = 1$ for all the other tasks. The coefficients for the global compactness term are set as $\alpha_c = \frac{N_S}{n_S^c}, \beta_c = \frac{N_T}{n_T^c}$ such that the sizes of the classes are balanced. Similarly, the coefficients for the local neighborhood term are such that $\alpha_c = \frac{N_S}{k+k^i}, \beta_c = \frac{N_T}{k+k^i}$, where k and k^i are defined in Section V-A-2). The influence of the hyper-parameters will be studied in Section VI-D. In GEF-LMNN and GEF-MFA, the numbers of nearest neighbors of interest are all set to be 3, i.e., $k = k_1 = k_2 = 3$. The iteration number is fixed to 10. for DTSL, the results in [43] are used. RTML and OT-GL are domain adaptation algorithms that are not within our framework, and the parameters are independently selected. In RTML, MMD is also used, but aims to discover a more robust feature space instead of a more compact one. We conduct the training according to [8]. For OT-GL, we implement the algorithm provided in [15]. SCA can essentially be interpreted using our framework, but has several different hyper-parameters. We directly adopt the experimental results reported in [17].

C. Experimental Results

Classification accuracies of the four proposed models and other baselines on the cross-domain tasks are reported in Table II, III and IV.

The results on hand-written digit and object datasets are summarized in Table II and III. In Table II are listed the results on low dimensional features. In general, our proposed models perform better than the other baselines, among which GEF-LDA has the best average accuracy. On hand-written digit datasets, local neighborhood strategies obtain better results than other baselines. On object datasets, our models achieve the best in 6 out of 12 tasks. Comparing the results with that of JDA and TJM, we can manifest that the added term can significantly improve the classifier trained from the learned features. RTML and OT-GL are two main competitive baselines.

TABLE IV
ACCURACY(%) ON CMU-PIE DATABASE

Tasks	1NN	JDA	DTSL	RTML	OT-GL	DICD	GEF-PCA	GEF-LDA	GEF-LMNN	GEF-MFA
PIE05→PIE07	26.09	58.81	65.87	60.12	64.89	72.99	**84.16**	81.28	74.83	72.74
PIE05→PIE09	26.59	54.23	64.09	55.21	71.94	72.00	73.47	**77.33**	53.55	64.40
PIE05→PIE27	30.67	84.50	82.03	85.19	85.25	92.22	**95.88**	93.96	90.30	92.13
PIE05→PIE29	16.67	49.75	54.90	52.98	60.42	**66.85**	48.16	55.39	40.32	48.65
PIE07→PIE05	24.49	57.62	45.04	58.13	60.14	69.93	**78.84**	73.71	69.54	72.54
PIE07→PIE09	46.63	62.93	53.49	63.92	74.20	65.87	73.84	**76.53**	70.77	73.96
PIE07→PIE27	54.07	75.82	71.43	76.16	78.10	85.25	**90.30**	89.19	87.83	89.43
PIE07→PIE29	26.53	39.89	47.97	40.38	58.21	48.71	**67.10**	64.15	47.12	57.78
PIE09→PIE05	21.37	50.96	52.49	53.12	61.34	**69.36**	62.00	69.30	58.46	57.11
PIE09→PIE07	41.01	57.95	55.56	58.67	73.48	65.44	73.05	70.66	59.67	**76.12**
PIE09→PIE27	46.53	68.45	77.50	69.81	80.99	83.39	83.93	86.06	**87.32**	87.08
PIE09→PIE29	26.23	39.95	54.11	42.13	66.97	61.40	63.05	**69.30**	52.76	55.27
PIE27→PIE05	32.95	80.58	81.54	81.12	87.70	93.13	93.31	**93.88**	92.56	86.22
PIE27→PIE07	62.68	82.63	85.39	83.92	89.69	90.12	**94.97**	94.48	92.45	76.86
PIE27→PIE09	73.22	87.25	82.23	89.51	89.89	88.97	92.65	91.97	91.73	**93.81**
PIE27→PIE29	37.19	54.66	72.61	56.26	79.17	75.61	78.80	**81.92**	64.77	64.34
PIE29→PIE05	18.49	46.46	52.19	29.11	48.11	**62.88**	45.29	55.67	55.22	60.44
PIE29→PIE07	24.19	42.05	49.41	33.28	53.04	57.03	**66.48**	64.46	52.67	59.61
PIE29→PIE09	28.31	53.31	58.45	39.85	67.95	65.87	71.69	**74.26**	66.18	59.07
PIE29→PIE27	31.24	57.01	64.31	47.13	64.46	74.77	77.14	78.64	**79.81**	73.24
Average	34.76	60.24	63.53	58.80	70.80	73.09	75.71	**77.11**	69.39	71.04

They also outperform JDA, TJM and SCA significantly. Notice that neither two algorithms use MMD distance typically. RTML implements low-rank transfer metric learning, while OT-GL replaces MMD with the earth mover's distance. This suggests that the low-rank strategy and earth mover's distance are competitive with MMD distance for the dataset with SRUF-BoW features.

In Table III are the results on Office+Caltech256 with DeCAF_6 features. The results on the DeCAF_6 features are better than on SURF features to a great extent, verifying that deep representations can bridge the domain gap. Our models outperform other baseline methods in 10 out of 12 tasks. An interesting observation is that these best accuracies are acquired by different models, manifesting that within the proposed framework, there always exists one model that is more suitable to deal with certain type of datasets than others. Although SCA is structurally similar to our framework, it fails to perform competitively on all the datasets. This is because during the training process, SCA does not utilize the label information and the predicted pseudo target labels iteratively like the proposed algorithms do.

The results on CMU-PIE face database are summarized in Table IV. The PCA and LDA based models outperform all the other baselines significantly, and have the highest classification accuracy rates on most of the domain pairs. To be specific, GEF-PCA gains a performance improvement of 25.68%, and GEF-LDA performs 28.00% higher, compared to the classic baseline JDA. Compared to the best baseline DICD, GEF-LDA still improves by 5.50%. DICD performs the best on 3 tasks. It is worth noting that, DICD is essentially within our framework. Its results are slightly worse than GEF-PCA probably because there are too many terms in the object function, and the balance among these terms are not easy to achieve.

We particularly compare the satisfying results of our models on human face database and those on the object SURF feature dataset in Table II. A possible explanation on why our models behave quite differently is that, a graph embedding algorithm behaves better on datasets containing more manifold information. For the object SURF feature dataset, since SURF features do not contain sufficient manifold information, effect of the graph embedding framework is weakened.

In conclusion, the experimental results demonstrate that, the proposed revised object function successfully improves MMD-based feature representation learning, and that the algorithms within our framework are capable of classifying cross-domain samples appropriately, thereby verifying the effectiveness of the framework.

D. Analytical Experiments

In this section, some analytical experiments are conducted to intuitively and visually illustrate the proposed framework. The framework's capability can be further verified in these experiments.

1) Features Visualization: Fig. 7 illustrates how the t-SNE [32] features for task MNIST→USPS extracted from the MMD-based algorithms are distributed. On the top row are the visualization plots for raw sample features and JDA features, and the middle and bottom are for feature representations learned from the models designed within the proposed framework. Each color represents a class with source and target samples mixed. From the plots, we have several observations. Firstly, these plots manifest that, MMD-based feature representation learning algorithms succeed in matching the source and target domains. Secondly, the samples with JDA features are not well separated according to the labels. The distributions of several classes prominently overlap due to larger within-class scatter in plot (b), but are distinctly separated by raw

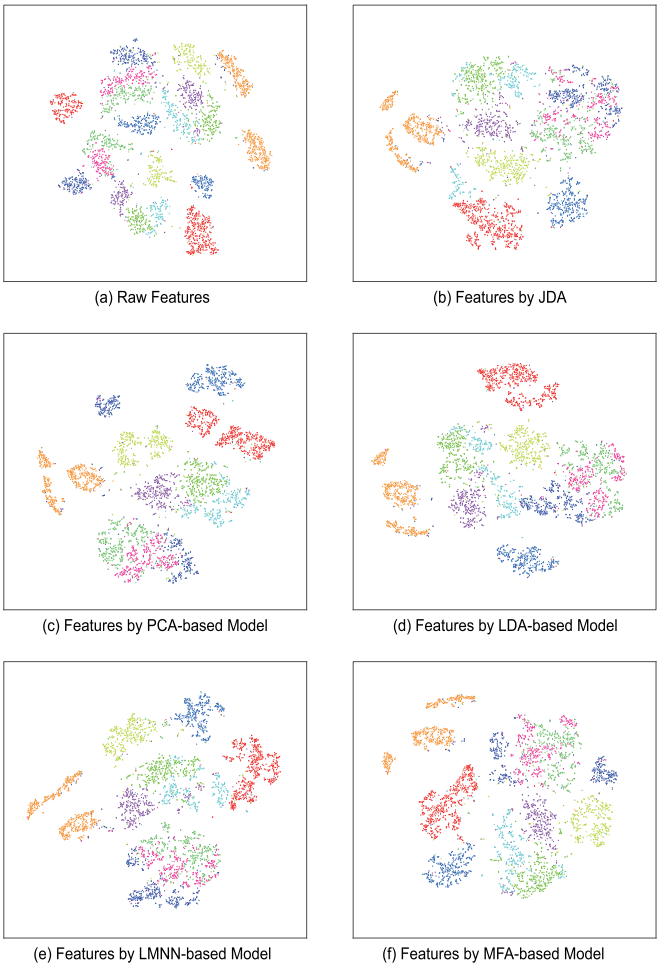


Fig. 7. Data embedding by t-SNE. Both the source and target points are plotted. Each color represents a class.

features in plot (a). The phenomenon strongly confirms our analysis in Section III. The last observation is that, the sample points in the local neighborhood based models (on the bottom row) are not distributed as tightly within each class as those in the global compactness based models (on the middle row). This is because they only focus on local structure. But different classes are still separated more clearly by the LMNN and MFA based models than by JDA. In conclusion, the visual plots demonstrate that our framework is advantageous in class clustering for domain adaptation problems.

2) Similarity of Embeddings: We directly show the similarity matrices of the embeddings of various methods to illustrate the effect of feature learning, as shown in Fig. 8. In the similarity matrix, the top-left and bottom-right submatrices indicate source and target domain similarity, respectively, and the top-right and bottom-left submatrices are cross-domain similarity. Correspondingly, the diagonal blocks of the submatrices indicate within-class similarity within and across domains.

It can be seen that models with revised graphs can extract better embeddings on which inter-class similarity is lower than JDA. This proves that the revision is able to improve the quality of the features learned through MMD minimization. It is worth noticing that the within-class similarity appears lower in GEF-MFA embedding than on GEF-LDA embedding. This is consistent with the mechanism of MFA algorithm, which is pulling *local* neighbors nearer rather than global same-class points.

The q values beneath the matrices indicate the proportions of elements in each matrix whose similarity matches the ground-truth label relations. It can be seen that models within our framework significantly outperform classic JDA.

3) Convergence Property: We empirically study the convergence property of the proposed models. The MMD distance on task PIE05→PIE07 and the classification accuracies on task MNIST→USPS are shown in Fig. 9. The empirical MMD values are calculated using true target labels. Fig. 9(a) shows that all the feature representation learning methods can reduce marginal and conditional MMD distances by iteratively refining the pseudo labels, and the distribution differences converge within 10 iterations. But the methods with the revised object function can reach smaller MMD than the simple JDA, verifying the effectiveness of the framework. Fig. 9(b) depicts the trends of classification accuracies of the methods with respect to the number of iterations. Generally, algorithms designed from the proposed framework have higher accuracies than JDA, which has been shown in Table II. The accuracy of GEF-PCA keeps increasing in the 20 iterations, while the accuracy of GEF-LDA converges within only 7 iterations. Accuracies of methods with the local neighborhood strategy fluctuate in a small range, probably caused by iterative selection of nearest neighbors. An interesting phenomenon is that the accuracy of JDA rises in the first four iterations and sharply drops in the next three iterations. The reason may lie in the unstable change of pseudo target labels used in JDA.

4) Parameter Sensitivity: Two tunable hyper-parameters are analyzed in the proposed framework simultaneously. One is λ, the trade-off parameter for the regularizer. The other one is the edge weights of the revision graph. As is stated in Section VI-A, the respective weights for source and target domains, α_c and β_c, are empirically fixed in the classification experiments. To conduct the parameter sensitivity analysis, we uniformly multiply α_c and β_c by a coefficient κ and change the object function as

$$\mathrm{Tr}\left[L^{\mathrm{T}}\left(X(\sum_{c=0}^{C}M^c+\kappa R)X^{\mathrm{T}}+\lambda I\right)L\right].$$

Then we only need to study the influence of the multiplier κ.

The results are shown in Fig. 10. We could find out that moderate λ and κ yield better performance. A large λ or κ may weaken the effectiveness of MMD. A small λ will result in overfitting, while a small κ leads to the performance of JDA. The results show that both types of our added terms play an important role in the feature learning.

Inspired by the works on ranking comparison [10]–[12], we consider how many same-class points are within the K-neighborhood. We gradually increase the value of K and implement K-nearest-neighbor classification (KNN). We have compared the accuracies of KNN, JDA, GEF-LDA and GEF-MFA with K ranging from 1 to 10 on two tasks. The results

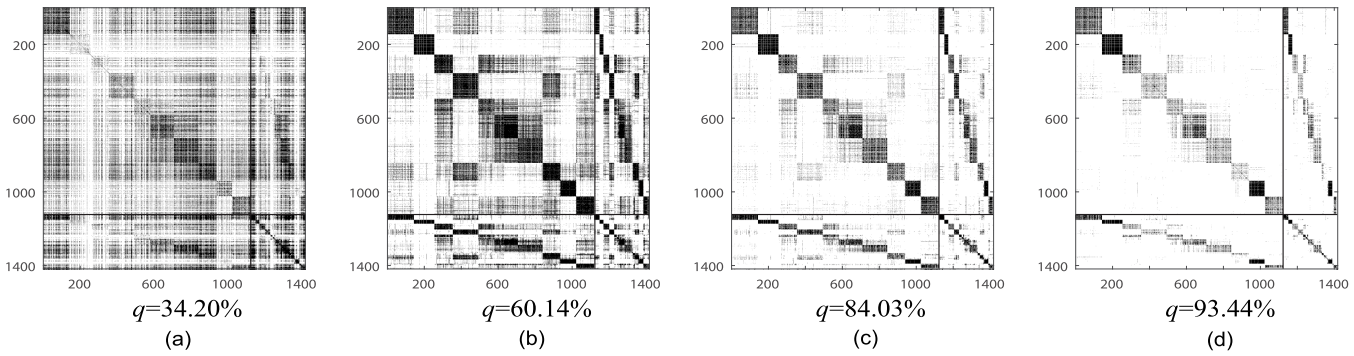


Fig. 8. Similarity matrix of embeddings on DeCAF$_6$ C→W task. The matrix has been sorted according to the classes for clearer visualization. (a) Similarity of raw feature embedding. (b) Similarity of JDA embedding. (c) Similarity of GEF-LDA embedding. (d) Similarity of GEF-MFA embedding.

TABLE V
ACCURACY(%) FOR MULTI-SOURCE TASKS

Source (PIE)	Target (PIE)	Global Compactness		Local Neighborhood	
		GEF-PCA	GEF-LDA	GEF-LMNN	GEF-MFA
05+07+09+27	29	58.58	56.56	62.32	**62.99**
05+07+09+29	27	84.92	84.98	**89.79**	86.54
05+07+27+29	09	80.51	79.90	81.25	**81.56**
05+09+27+29	07	78.76	76.43	**82.44**	71.21
07+09+27+29	05	57.83	66.96	**85.38**	58.82

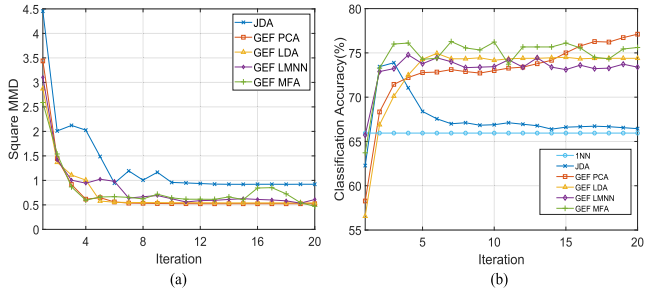


Fig. 9. Convergence properties of the four proposed models and JDA. (a) MMD distance w.r.t number of iterations on PIE05→PIE07 task. (b) Classification accuracy w.r.t number of iterations on MNIST→USPS task.

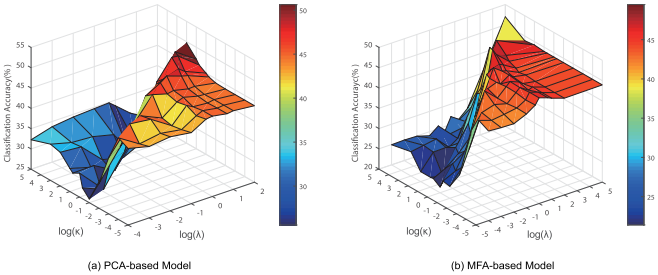


Fig. 10. Parameter sensitivity study of λ and κ with GEF-PCA and GEF-MFA on task C→A. The logarithm of the parameters is used to rescale the size.

are shown in Fig. 11. We notice that in general, models within our framework perform more stably than KNN and JDA, indicating that for each data point, points of other classes are rare in the neighborhood and are ranked relatively backward with the features of our models.

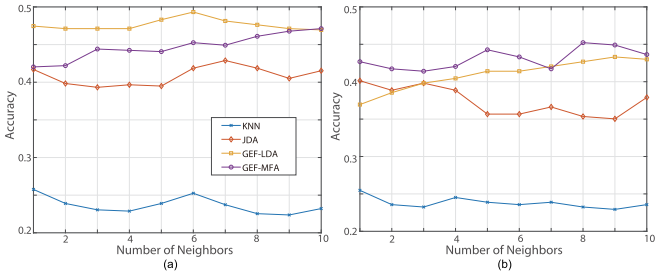


Fig. 11. Parameter sensitivity study of K while implementing KNN classification. (a) C→ W SURF task. (b) A→ D SURF task.

5) Adaptive Coefficients: In all the experiment setting above, the coefficients α_c and β_c only depend on the class of the sample pair, and are *fixed* within each class. We make a further study on adapting the coefficients for term $\|A^{\mathrm{T}}(x_i - x_j)\|^2$ according to the distance between the pair in the original space. Two strategies of coefficient adaptation are proposed:

$$\textit{adaptive+}: (\alpha_c, \beta_c) \rightarrow (\alpha_c, \beta_c)\|x_i - x_j\|^2$$
$$\textit{adaptive-}: (\alpha_c, \beta_c) \rightarrow \frac{(\alpha_c, \beta_c)}{\|x_i - x_j\|^2}.$$

The symbols "+" and "−" indicate the coefficients are adapted positively and negatively relevant to the distance, respectively.

We run GEF-PCA and GEF-LDA on dataset PIE, and plot classification accuracies with three coefficient settings in Fig. 12. Generally, the order of performance is: *adaptive+* > *fixed* > *adaptive-*. This is a reasonable conclusion because by adapting coefficients in *adaptive+*, the pairs that are not

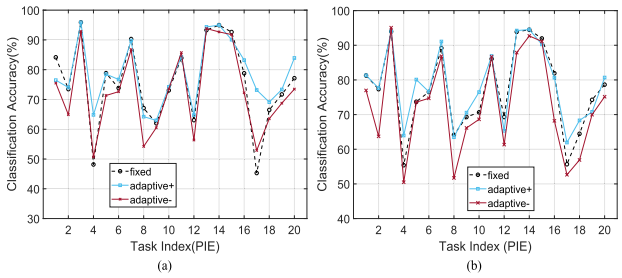


Fig. 12. Classification accuracies on dataset PIE using different coefficient adaptation strategies. (a) Performance of GEF-PCA. (b) Performance of GEF-LDA.

similar in the same class are paid more attention, while in *adaptive-*, such pairs are less focused.

6) Model Choosing Strategy: Extensive experiments are carried out on several multi-source tasks to verify a model choosing strategy: local neighborhood strategy is more suitable for multimodal distributed data. We artificially make a dataset by selecting 1/6 of the data from 4 PIE subsets respectively to form a multi-source domain. Then we adapt this domain to the other one PIE subset using our models.

Cross-domain classification accuracies are shown in Table V. Models adopting local neighborhood strategy have higher accuracies for all these tasks with less clustered data. GEF-LMNN is better than GEF-MFA because it considers margins between classes other than neighbor distances. Notice that on single source tasks in Table IV, GEF-LMNN and GEF-MFA have the best results for only 4 tasks, proving that the performance in Table V is highly related to the multimodal distributions of source domains.

VII. Conclusion

In this paper, we aim to provide insights into the MMD metric from the point of view of graph embedding. Some unapparent properties are discovered. We prove that minimizing MMD distance equals to a graph embedding process in which each domain becomes more scattered. To cope with this problem, a revision on the intrinsic graph of the empirical MMD minimization regarding within-domain compactness is necessary for MMD-based domain adaptation algorithms, and the two terms should be treated as one part in the object function. A graph embedding framework for MMD-based domain adaptation algorithms has been presented to enhance within-domain compactness while reducing between-distribution distance. The framework can be used to develop feature representation learning models focusing on global or local sample distribution. Two revision strategies and four models within the framework are explored. We give a model choosing strategy and empirically prove it. Within the framework, models can be put forward for specific cases.

A possible extension of our work is the direct graph embedding algorithm design for domain adaptation. Inspired by the MMD method, we may be able to directly design intrinsic graphs for domain adaptation by appropriately assigning similarity weights. We intend to further investigate the graph embedding issue in the area of domain adaptation in both theory and practice.

References

[1] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 769–776.

[2] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 585–591.

[3] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[4] M. Brand, "Continuous nonlinear dimensionality reduction by kernel eigenmaps," in *Proc. IJCAI*, 2003, pp. 547–554.

[5] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, Sep. 2017.

[6] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[7] Z. Ding, N. M. Nasrabadi, and Y. Fu, "Semi-supervised deep domain adaptation via coupled neural networks," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5214–5224, Nov. 2018.

[8] Z. Ding and Y. Fu, "Robust transfer metric learning for image classification," *IEEE Trans. Image Process.*, vol. IT-26, no. 2, pp. 660–670, Feb. 2017.

[9] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.

[10] X. Dong and M. J. Chantler, "Perceptually motivated image features using contours," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5050–5062, Nov. 2016.

[11] X. Dong and J. Dong, "The visual word booster: A spatial layout of words descriptor exploiting contour cues," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3904–3917, Aug. 2018.

[12] X. Dong, T. S. Methven, and M. J. Chantler, "How well do computational features perceptually rank textures? A comparative evaluation," in *Proc. Int. Conf. Multimedia Retr.*, 2014, p. 281.

[13] L. Duan, I. W. Tsang, and D. Xu, "Domain transfer multiple kernel learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 465–479, Mar. 2012.

[14] R. A. Fisher, "The use of multiple measures in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[15] R. Flamary and N. Courty, "POT Python optimal transport library," 2017. [Online]. Available: https://github.com/rflamary/POT

[16] B. Geng, D. Tao, and C. Xu, "DAML: Domain adaptation metric learning," *IEEE Trans. Image Process.*, vol. 20, no. 10, pp. 2980–2989, Oct. 2011.

[17] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter component analysis: A unified framework for domain adaptation and domain generalization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1414–1430, Jul. 2017.

[18] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2066–2073.

[19] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl. Based Syst.*, vol. 151, pp. 78–94, Jul. 2018.

[20] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 513–520.

[21] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CaltechAUTHORS:CNS-TR-2007-001, 2007.

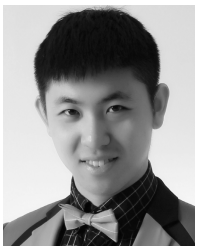
[22] C.-A. Hou, Y.-H. H. Tsai, Y.-R. Yeh, and Y.-C. F. Wang, "Unsupervised domain adaptation with label and structural consistency," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5552–5562, Dec. 2016.

[23] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 601–608.

[24] M. Jiang, W. Huang, Z. Huang, and G. G. Yen, "Integration of global and local metrics for domain adaptation learning via dimensionality reduction," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 38–51, Dec. 2017.

[25] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal Component Analysis*. Berlin, Germany: Springer, 1986, pp. 115–128.

[26] S. Li, S. Song, and G. Huang, "Prediction reweighting for domain adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1682–1695, Jul. 2016.
[27] S. Li, S. Song, G. Huang, Z. Ding, and C. Wu, "Domain invariant and class discriminative feature learning for visual domain adaptation," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4260–4273, Sep. 2018.
[28] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1134–1148, Jun. 2013.
[29] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2200–2207.
[30] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1410–1417.
[31] D. Luo, F. Nie, H. Huang, and C. H. Ding, "Cauchy graph embedding," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 553–560.
[32] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[33] A. M. Martínez and A. C. Kak, "PCA versus LDA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, Feb. 2001.
[34] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction," in *Proc. AAAI*, vol. 8, 2008, pp. 677–682.
[35] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.
[36] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[37] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, Dec. 2000.
[38] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 213–226.
[39] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, May 2014.
[40] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," 2017, *arXiv:1707.01217*. [Online]. Available: https://arxiv.org/abs/1707.01217
[41] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression (PIE) database," in *Proc. IEEE 5th Int. Conf. Autom. Face Gesture Recognit.*, May 2002, pp. 53–58.
[42] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, Feb. 2009.
[43] Y. Xu, X. Fang, J. Wu, X. Li, and D. Zhang, "Discriminative transfer subspace learning via low-rank and sparse representation," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 850–863, Feb. 2016.
[44] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[45] J. Yang, S. Yang, Y. Fu, X. Li, and T. Huang, "Non-negative graph embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[46] H. Zhang, Z.-J. Zha, S. Yan, M. Wang, and T.-S. Chua, "Robust non-negative graph embedding: Towards noisy data, unreliable graphs, and noisy labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2464–2471.



Yiming Chen received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree with the Institute of System Integration. His current research interests include pattern recognition and machine learning, especially in transfer learning and domain adaptation.

Shiji Song received the Ph.D. degree from the Department of Mathematics, Harbin Institute of Technology, in 1996. He is currently a Professor with the Department of Automation, Tsinghua University. His research interests include system modeling, control and optimization, computational intelligence, and pattern recognition.

Shuang Li received the B.S. degree from the Department of Automation, Northeastern University, in 2012, and the Ph.D. degree from the Institute of System Integration, Department of Automation, Tsinghua University, in 2018. He was a Visiting Research Scholar with the Department of Computer Science, Cornell University from 2015 to 2016. He is currently an Assistant Professor with the School of Computer Science and Technology, Beijing Institute of Technology, China. His main research interests include machine learning and pattern recognition, especially in transfer learning and domain adaptation.

Cheng Wu received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1962 and 1966, respectively.

Since 1967, he has been with Tsinghua University, where he is currently a Professor with the Department of Automation. His current research interests include system integration, modeling, scheduling, and optimization of complex industrial systems.

Prof. Wu is a member of the Chinese Academy of Engineering.