Neurocomputing 412 (2020) 115–128

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Joint distribution matching embedding for unsupervised domain adaptation

Xiaona Jin [a], Xiaowei Yang [a], Bo Fu [a], Sentao Chen [a,*]

[a] School of Software Engineering, South China University of Technology, Guangzhou 510006, China

ARTICLE INFO

Article history:
Received 21 November 2019
Revised 16 April 2020
Accepted 31 May 2020
Available online 12 June 2020
Communicated by Jungong Han

Keywords:
Domain adaptation
Joint distribution
Maximum mean discrepancy
Grassmann manifold

ABSTRACT

When the distributions between the source (training) and target (test) datasets are different, the performance of classical statistical learning methods degrades significantly. Domain adaptation (DA) aims at correcting this distribution mismatch and narrowing down the distribution discrepancy. Existing methods mostly focus on correcting the mismatch between the marginal distributions and/or the class-conditional distributions. In this paper, we assume that the distribution mismatch in domain adaptation is the joint distribution mismatch, and propose an Extended Maximum Mean Discrepancy (EMMD) metric to measure the distance between joint distributions. Based on this metric, we propose the Joint Distribution Matching Embedding (JDME) approach, which finds a mapping matrix to project the samples into a latent space, where the EMMD between the source and target joint distributions is minimized. The resultant orthogonal-constrained optimization problem can be solved in the form of an unconstrained problem on the Grassmann manifold. After the joint distribution matching, we can expect the classical statistical learning methods to perform well on the target dataset. Experiments on object recognition, face recognition, and spam filtering demonstrate that our method statistically outperforms the state-of-the-art shallow methods and is also on par with the deep learning methods.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

There is usually a distribution difference between the source (training) and target (test) datasets. Reasons accounting for the discrepancy are various and application-dependent. In computer vision, it may result from a different device, illumination or angle. While in text analysis, it may attribute to a different time or context. Since most classical statistical learning methods are built on the independent-and-identically-distributed (i.i.d.) assumption, the existence of the distribution difference makes their performance decrease significantly. Domain adaptation (DA) exactly aims at correcting this distribution mismatch and narrowing down the distribution gap. Recent years have witnessed the successful application of DA in various fields, such as object recognition [15,22,32,36,39,59], face recognition [11,22,37], text categorization [3,7,59], sentiment analysis [26,27,43,49], and indoor WiFi localization [3,44].

Depending on whether the labeled data are available in the target domain, DA is divided into unsupervised domain adaptation (UDA) and semi-supervised domain adaptation (SDA). UDA and SDA uniformly hold abundant labeled data in the source domain. In the target domain, the unlabeled data is available for both UDA and SDA, and SDA additionally requires a small fraction of labeled data. In practice, sometimes it may cost a lot of manpower and time to obtain the labels for the target data. Considering this and our interest as well, we will focus on UDA in this paper.

A number of various methods have been proposed for addressing the UDA problem. According to the learning mechanism, these adaptation methods can be roughly categorized into instance reweighting adaptation, feature adaptation, classifier adaptation, and neural network adaptation.

Instance reweighting adaptation reweights the source samples so as to reduce the distribution discrepancy between the two domains [8,20,24,30,55]. Kernel Mean Matching (KMM) [24] reweights the source data such that the discrepancy between the kernel means of the reweighted source samples and the target samples is minimized. Feature adaptation, as a more popular methodology, focuses on seeking a latent space where the domain discrepancy is minimized [3,14,28,32,37,57]. Specifically, feature adaptation can be further categorized into distribution matching [3,11,27,28,31,32,37,57], subspace learning [14], covariance matching [49,59], and more. For distribution matching, it aims at minimizing the statistical distances between domains to learn

* Corresponding author.
E-mail address: sentaochen@yahoo.com (S. Chen).

https://doi.org/10.1016/j.neucom.2020.05.098
0925-2312/© 2020 Elsevier B.V. All rights reserved.

the feature mapping (mappings). For example, Joint Distribution Adaptation (JDA) [37] minimizes the Maximum Mean Discrepancy (MMD) [19] between the source and target distributions, and the MMD between the source and target class-conditional distributions to learn a mapping matrix. Building on JDA, Domain-Irrelevant Class Clustering (DICE) [32] further promotes class clustering for both the source and target data, while Domain Invariant and Class Discriminative feature learning (DICD) [31] considers both the intra-class and inter-class scatter when jointly matching the distributions. To better improve the effect of distribution matching, the important class imbalance problem [53] and the different contributions of the marginal and conditional distributions to the domain discrepancy [52,54] are also considered in this line of works. For subspace learning, it intends to match the subspaces spanned by the eigenvectors of the source and target samples [14]. For covariance matching, it matches the source and target covariance matrices in the Euclidean space [49] or the Reproducing Kernel Hilbert Space (RKHS) [59]. Classifier adaptation trains a classifier on the labeled source data and unlabeled target data, and uses it to predict labels for the target data [4,7,33]. Liu et al. [33] proposed the Structure-Preserved Unsupervised Domain Adaptation (SP-UDA) framework, which ensures that the whole structure of the source data is preserved so as to guide the target structure learning in a semi-supervised clustering manner.

Different from the shallow approaches mentioned above, neural network adaptation integrates feature extraction with knowledge transfer in an end-to-end model. It can be roughly divided into discrepancy-based methods [35,38,45] and adversarial-based methods [9,15,29,36,58]. The discrepancy-based methods mainly minimize the MMD metric, whereas the adversarial-based methods minimize the domain difference via an adversarial objective. As a discrepancy-based method, Joint Adaptation Network (JAN) [38] minimizes the Joint Maximum Mean Discrepancy (JMMD) between the source and target joint distributions of the full-connection activations from multiple layers of a network. For adversarial-based methods, Domain-Adversarial Neural Network (DANN) [15] intends to learn a domain-invariant but class-discriminant feature representation in an adversarial way. To better match the distributions, Li et al. [29] introduced a Joint Adversarial Domain Adaptation (JADA) approach, which jointly matches the domain-wise and class-wise distributions across the source and target in a unified adversarial learning process. Zhang et al. [58] proposed a new divergence named Margin Disparity Discrepancy for distribution comparison, and designed an adversarial learning algorithm based on this divergence. Generally, the neural network adaptation approaches can present a better performance on large datasets due to the expressive neural network structure.

However, almost all these approaches focus on matching the marginal distributions and/or the class-conditional distributions between the source and target domains. In this paper, we assume that the distribution mismatch in domain adaptation is the joint distribution mismatch. To measure the distance between joint distributions, an Extended Maximum Mean Discrepancy (EMMD) metric is proposed. The metric extends the kernel function of the Maximum Mean Discrepancy (MMD) [19] to a product kernel of the feature and the label. Using this metric, we propose a simple yet effective method named Joint Distribution Matching Embedding (JDME). The proposed method seeks a mapping matrix to project the samples into a latent space, where the EMMD between the source and target joint distributions is minimized. In particular, we optimize the objective function by constraining the matrix variable as an orthogonal matrix, which has been well practiced in previous domain adaptation works [3,22]. From a geometrical viewpoint, this constrained problem can be regarded as an unconstrained one on the Grassmann manifold [13]. Extensive experiments are conducted to demonstrate the benefits of our method. The main contributions of this paper are summarized as follows:

1) An Extended Maximum Mean Discrepancy (EMMD) metric is proposed to measure the distance between joint distributions of the feature and the label.
2) A simple yet effective method named JDME is proposed to address the UDA problem by learning a projection matrix so that the joint distributions are similar between the two domains.
3) Extensive experiments on real-world image and text datasets show the superiority of the proposed method.

The remainder of this paper is organized as follows: Section 2 introduces the problem formulation, the EMMD metric, the JDME model and the corresponding optimization algorithm. In Section 3, we conduct extensive experiments to evaluate the proposed method and compare it with the state-of-the-art UDA methods. Section 4 concludes this paper and indicates possible research directions in the future.

2. Proposed approach

In this part, we first introduce the formulation of UDA. Following that, the EMMD metric is proposed to measure the distance between joint distributions. On this basis, the JDME is modeled to learn a mapping that alleviates the joint distribution gap. Finally, we solve the resultant optimization problem to learn the mapping.

2.1. Problem formulation

We denote $\boldsymbol{x}_i \in \mathbb{R}^{D\times 1}$ as a D-dimensional feature vector, and $y_i \in \{1, 2, \ldots, C\}$ as the corresponding label. In UDA, we have n_s labeled source samples $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and n_t unlabeled target samples $\mathcal{D}^u = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$. The source samples are drawn from the source joint distribution $P_{XY}^s(\boldsymbol{x}, y)$, i.e. $(\boldsymbol{x}_i^s, y_i^s) \sim P_{XY}^s(\boldsymbol{x}, y)$, and the unlabeled target samples are drawn from the target distribution $P_X^t(\boldsymbol{x})$, i.e. $\boldsymbol{x}_i^t \sim P_X^t(\boldsymbol{x}) = \int_y P_{XY}^t(\boldsymbol{x}, y)dy$, where $P_{XY}^t(\boldsymbol{x}, y)$ is the target joint distribution. Under the joint distribution mismatch $P_{XY}^s(\boldsymbol{x}, y) \neq P_{XY}^t(\boldsymbol{x}, y)$, UDA intends to adapt the joint distributions and generalize a learning machine from the source domain to the target domain.

2.2. Extended maximum mean discrepancy

We first briefly review the existing MMD [19], and then extend it to the EMMD metric. The main idea of MMD is to find a function such that the difference between the function expectation values with respect to two distributions is the largest. To be specific, it is defined as

$$\mathrm{MMD}(\mathcal{F}, P_X^s, P_X^t) = \sup_{f\in\mathcal{F}}\Big(E_{\boldsymbol{x}^s\sim P_X^s}[f(\boldsymbol{x}^s)] - E_{\boldsymbol{x}^t\sim P_X^t}[f(\boldsymbol{x}^t)]\Big), \quad (1)$$

where P_X^s and P_X^t are the source and target distributions, $E_{\boldsymbol{x}\sim P_X}[\cdot]$ is the mathematical expectation with respect to P_X, and $\mathcal{F}$ represents a set of functions in the unit ball in the RKHS $\mathcal{H}$. According to [19], the empirical estimate of MMD is calculated as the difference between the source and target sample means in the RKHS:

$$\begin{aligned}\widehat{\mathrm{MMD}}(\boldsymbol{X}^s, \boldsymbol{X}^t) &= \left\| \frac{1}{n_s}\sum_{i=1}^{n_s}\phi(\boldsymbol{x}_i^s) - \frac{1}{n_t}\sum_{i=1}^{n_t}\phi(\boldsymbol{x}_i^t)\right\|_{\mathcal{H}} \\ &= \left(\frac{1}{n_s^2}\sum_{i,j=1}^{n_s}k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s) + \frac{1}{n_t^2}\sum_{i,j=1}^{n_t}k(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t) - \frac{2}{n_s n_t}\sum_{i,j=1}^{n_s,n_t}k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^t)\right)^{\frac{1}{2}}, \quad (2)\end{aligned}$$

where $\boldsymbol{X}^s = [\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_{n_s}^s] \in \mathbb{R}^{D\times n_s}$ and $\boldsymbol{X}^t = [\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_{n_t}^t] \in \mathbb{R}^{D\times n_t}$ represent the source and target samples drawn from $P_X^s(\boldsymbol{x})$ and $P_X^t(\boldsymbol{x})$, respectively. Besides, $\phi(\cdot)$ is a mapping to the RKHS $\mathcal{H}$, and $k(\cdot,\cdot) = \langle\phi(\cdot), \phi(\cdot)\rangle$ is the corresponding kernel function.

Apparently, MMD only measures the distance between marginal distributions, here we further extend it to measure the distance between joint distributions $P_{XY}^s(\boldsymbol{x}, y)$ and $P_{XY}^t(\boldsymbol{x}, y)$, and propose the Extended Maximum Mean Discrepancy (EMMD) metric as follows:

$$\mathrm{EMMD}(\mathcal{G}, P_{XY}^s, P_{XY}^t) = \sup_{g\in\mathcal{G}}\Big(E_{(\boldsymbol{x}^s,y^s)\sim P_{XY}^s}[g(\boldsymbol{x}^s, y^s)] - E_{(\boldsymbol{x}^t,y^t)\sim P_{XY}^t}\big[g(\boldsymbol{x}^t, y^t)\big]\Big), \tag{3}$$

where $\mathcal{G}$ represents a set of functions in the unit ball in the RKHS $\tilde{\mathcal{H}}$.

In practice, we need to estimate EMMD from discrete observations. Analogous to the estimate of MMD, given source samples $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and target samples $\mathcal{D}^t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t}$ respectively drawn from $P_{XY}^s(\boldsymbol{x}, y)$ and $P_{XY}^t(\boldsymbol{x}, y)$, the empirical estimate of EMMD is expressed as

$$\begin{aligned}\widehat{\mathrm{EMMD}}(\mathcal{D}^s, \mathcal{D}^t) &= \left\|\frac{1}{n_s}\sum_{i=1}^{n_s}\varphi(\boldsymbol{x}_i^s, y_i^s) - \frac{1}{n_t}\sum_{i=1}^{n_t}\varphi(\boldsymbol{x}_i^t, y_i^t)\right\|_{\tilde{\mathcal{H}}} \\ &= \left(\frac{1}{n_s^2}\sum_{i,j=1}^{n_s}\sigma\left[(\boldsymbol{x}_i^s, y_i^s), (\boldsymbol{x}_j^s, y_j^s)\right] + \frac{1}{n_t^2}\sum_{i,j=1}^{n_t}\sigma\left[(\boldsymbol{x}_i^t, y_i^t), (\boldsymbol{x}_j^t, y_j^t)\right]\right. \\ &\quad \left. - \frac{2}{n_s n_t}\sum_{i,j=1}^{n_s,n_t}\sigma\left[(\boldsymbol{x}_i^s, y_i^s), (\boldsymbol{x}_j^t, y_j^t)\right]\right)^{\frac{1}{2}},\end{aligned} \tag{4}$$

where the kernel function $\sigma(\cdot,\cdot) = \langle\varphi(\cdot), \varphi(\cdot)\rangle$, and $\varphi(\cdot)$ is a mapping to the RKHS $\tilde{\mathcal{H}}$. Since the feature $\boldsymbol{x}$ and the label y play different roles in a classification or regression problem, we define the kernel function $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)]$ as a product kernel of $\boldsymbol{x}$ and y, namely $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)\tau(y_i, y_j)$, where $k(\cdot,\cdot)$ is the feature kernel for $\boldsymbol{x}$, and $\tau(\cdot,\cdot)$ is the label kernel for y. In the following proposition, we show that under certain condition, this product kernel is a well-defined kernel function.

Proposition 1. If a Gram matrix $\boldsymbol{T} \in \mathbb{R}^{(n_s+n_t)\times(n_s+n_t)}$ associated with the label kernel $\tau(y_i, y_j)$ can be expressed in the form $\boldsymbol{T} = \sum_{r=1}^R \boldsymbol{u}_r\boldsymbol{u}_r^\top$, with $\boldsymbol{u}_r$ being a (n_s+n_t)-dimensional vector, then for any feature kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)\tau(y_i, y_j)$ is still a kernel function.

Proof. Please see Appendix A.

In the next section, using Proposition 1 as a guideline, we design an appropriate label kernel $\tau(y_i, y_j)$ so as to make $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)\tau(y_i, y_j)$ a kernel function. With $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)]$ in this form, Eq. (4) can be rewritten as

$$\begin{aligned}\widehat{\mathrm{EMMD}}(\mathcal{D}^s, \mathcal{D}^t) = &\left(\frac{1}{n_s^2}\sum_{i,j=1}^{n_s}k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s)\tau(y_i^s, y_j^s) + \frac{1}{n_t^2}\sum_{i,j=1}^{n_t}k(\boldsymbol{x}_i^t, \boldsymbol{x}_j^t)\tau(y_i^t, y_j^t)\right. \\ &\left. - \frac{2}{n_s n_t}\sum_{i,j=1}^{n_s,n_t}k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^t)\tau(y_i^s, y_j^t)\right)^{\frac{1}{2}}.\end{aligned} \tag{5}$$

Later in Section 3.4.1, we experimentally show that in a classification problem where the label y is clearly dependent on its feature $\boldsymbol{x}$, our empirical metric in Eq. (5) is able to yield reasonable estimate of the distance between joint distributions.

Remark 1. Note that Long et al. [38] extended MMD, and proposed the Joint Maximum Mean Discrepancy (JMMD) to measure the distance between joint distributions. Specifically, using our notation, JMMD measures the distance between $P_{XY}^s(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^L)$ and $P_{XY}^t(\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^L)$, where $\boldsymbol{x}^i$ is the activations of the ith layer in a neural network. By contrast, EMMD computes the distance between $P_{XY}^s(\boldsymbol{x}, y)$ and $P_{XY}^t(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is the feature and y is the label for $\boldsymbol{x}$. Comparing their empirical estimate, we observe that the product kernel is also used in estimating JMMD. In particular, according to [38], its empirical estimate is expressed as

$$\begin{aligned}&\widehat{\mathrm{JMMD}}\left(\{(\boldsymbol{x}_i^{1s}, \boldsymbol{x}_i^{2s}, \ldots, \boldsymbol{x}_i^{Ls})\}_{i=1}^{n_s}, \{(\boldsymbol{x}_i^{1t}, \boldsymbol{x}_i^{2t}, \cdots, \boldsymbol{x}_i^{Lt})\}_{i=1}^{n_t}\right) \\ &= \left(\frac{1}{n_s^2}\sum_{i,j=1}^{n_s}\prod_{l=1}^{L}k^l(\boldsymbol{x}_i^{ls}, \boldsymbol{x}_j^{ls}) + \frac{1}{n_t^2}\sum_{i,j=1}^{n_t}\prod_{l=1}^{L}k^l(\boldsymbol{x}_i^{lt}, \boldsymbol{x}_j^{lt}) - \frac{2}{n_s n_t}\sum_{i,j=1}^{n_s,n_t}\prod_{l=1}^{L}k^l(\boldsymbol{x}_i^{ls}, \boldsymbol{x}_j^{lt})\right)^{\frac{1}{2}},\end{aligned} \tag{6}$$

where $\boldsymbol{x}_i^{ls}, \boldsymbol{x}_j^{lt}$ respectively represent the activations of the lth full connection layers from the ith source sample and the jth target sample, and $k^l(\boldsymbol{x}_i^{ls}, \boldsymbol{x}_j^{lt})$ is a kernel function. Obviously, the product kernel $k(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s)\tau(y_i^s, y_j^s)$ in Eq. (5) and the one $\prod_{l=1}^L k^l(\boldsymbol{x}_i^{ls}, \boldsymbol{x}_j^{ls}) = k^1(\boldsymbol{x}_i^{1s}, \boldsymbol{x}_j^{1s})\cdots k^L(\boldsymbol{x}_i^{Ls}, \boldsymbol{x}_j^{Ls})$ in Eq. (6) resemble each other.

2.3. Joint distribution matching embedding

We exploit the proposed EMMD metric to compare the source and target joint distributions. As mentioned earlier, it can be estimated from the labeled source and labeled target data. To obtain the labels for the target data in the UDA setting, we train an SVM classifier on the labeled source data and use it to predict the labels $\{\hat{y}_j^t\}_{i=1}^{n_t}$ for the unlabeled target data $\{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$. We then propose to search for a projection matrix $\boldsymbol{W} \in \mathbb{R}^{D\times d} (d \leqslant D)$ via minimizing the EMMD between the projected source data $\mathcal{D}_{\boldsymbol{W}}^s = \{(\boldsymbol{W}^\top\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ and the projected target data $\hat{\mathcal{D}}_{\boldsymbol{W}}^t = \{(\boldsymbol{W}^\top\boldsymbol{x}_i^t, \hat{y}_i^t)\}_{i=1}^{n_t}$. The resultant optimization problem is as follows

$$\begin{aligned}&\min_{\boldsymbol{W}}\widehat{\mathrm{EMMD}}^2(\mathcal{D}_{\boldsymbol{W}}^s, \hat{\mathcal{D}}_{\boldsymbol{W}}^t) \\ &\text{s.t. } \boldsymbol{W}^\top\boldsymbol{W} = \mathbf{I},\end{aligned} \tag{7}$$

where $\mathbf{I}$ is an $d\times d$ identity matrix. Here we enforce orthogonality constraints on $\boldsymbol{W}$, i.e. $\boldsymbol{W}^\top\boldsymbol{W} = \mathbf{I}$, since these constraints can not only avoid matrix degradation [3], but also keep the reconstructed features from redundancy [18]. According to Eq. (5), the objective function can be expressed as

$$\begin{aligned}\widehat{\mathrm{EMMD}}^2(\mathcal{D}_{\boldsymbol{W}}^s, \hat{\mathcal{D}}_{\boldsymbol{W}}^t) = &\frac{1}{n_s^2}\sum_{i,j=1}^{n_s}k(\boldsymbol{W}^\top\boldsymbol{x}_i^s, \boldsymbol{W}^\top\boldsymbol{x}_j^s)\tau(y_i^s, y_j^s) \\ &+ \frac{1}{n_t^2}\sum_{i,j=1}^{n_t}k(\boldsymbol{W}^\top\boldsymbol{x}_i^t, \boldsymbol{W}^\top\boldsymbol{x}_j^t)\tau(\hat{y}_i^t, \hat{y}_j^t) - \frac{2}{n_s n_t}\sum_{i,j=1}^{n_s,n_t}k(\boldsymbol{W}^\top\boldsymbol{x}_i^s, \boldsymbol{W}^\top\boldsymbol{x}_j^t)\tau(y_i^s, \hat{y}_j^t)\end{aligned} \tag{8}$$

This objective function actually is a class of functions, since there could be multiple choices for the feature kernel $k(\cdot,\cdot)$ and the label kernel $\tau(\cdot,\cdot)$, which makes the corresponding instances different from each other. In the following, we discuss the design for these two kernel functions. For the feature kernel, it can be chosen as the linear, polynomial, or Gaussian kernel, which are respectively defined as follows:

- Linear Kernel:

$$k(\boldsymbol{W}^\top x_i, \boldsymbol{W}^\top x_j) = \boldsymbol{x}_i^\top \boldsymbol{W}\boldsymbol{W}^\top \boldsymbol{x}_j, \tag{9}$$

- Polynomial Kernel:

$$k(\boldsymbol{W}^\top \boldsymbol{x}_i, \boldsymbol{W}^\top \boldsymbol{x}_j) = (\boldsymbol{x}_i^\top \boldsymbol{W}\boldsymbol{W}^\top \boldsymbol{x}_j + 1)^{deg}, \tag{10}$$

- Gaussian Kernel:

$$k(\boldsymbol{W}^\top \boldsymbol{x}_i, \boldsymbol{W}^\top \boldsymbol{x}_j) = \exp\left(-\frac{1}{\gamma}\|\boldsymbol{W}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2^2\right), \tag{11}$$

where deg is the degree of the polynomial kernel and γ is the scale parameter of the Gaussian kernel.

For the label kernel $\tau(\cdot,\cdot)$, it is defined as $\tau(\cdot,\cdot) = \tau_1(\cdot,\cdot) + \tau_2(\cdot,\cdot)$. $\tau_1(\cdot,\cdot)$ is a δ-kernel [16] which takes the following form

$$\tau_1(y_i, y_j) = \begin{cases} 1 & \text{if } y_i = y_j \\ 0 & \text{if } y_i \neq y_j \end{cases}. \tag{12}$$

And $\tau_2(\cdot,\cdot)$ is designed as

$$\tau_2(y_i, y_j) = \begin{cases} (n_s/n_s^c)^2 & \text{if } (y_i = y_j = c) \wedge (i,j \in \mathcal{S}) \\ (n_t/n_t^c)^2 & \text{if } (y_i = y_j = c) \wedge (i,j \in \mathcal{T}) \\ (n_s n_t)/(n_s^c n_t^c) & \text{if } (y_i = y_j = c) \wedge (i \in \mathcal{S} \wedge j \in \mathcal{T}) \\ (n_s n_t)/(n_s^c n_t^c) & \text{if } (y_i = y_j = c) \wedge (i \in \mathcal{T} \wedge j \in \mathcal{S}) \\ 0 & \text{if } y_i \neq y_j \end{cases}, \tag{13}$$

where n_s^c, n_t^c respectively represent the number of samples from the cth category in the source and target domains, and $\mathcal{S}, \mathcal{T}$ are two sets consisting of the indices of the source and target samples. The design of $\tau_2(\cdot,\cdot)$ is directly inspired from the works that are based on class-conditional distribution matching [28,32], and therefore we term this function as the class-conditional function. Moreover, we can easily verify that this function is a well-defined kernel function. The proof is provided in Appendix B. Since $\tau_1(\cdot,\cdot)$ and $\tau_2(\cdot,\cdot)$ are both kernel functions, their sum $\tau(\cdot,\cdot)$ is also a kernel function. More importantly, it can be proved that this kernel function $\tau(\cdot,\cdot)$ satisfies the condition in Proposition 1, making $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)\tau(y_i, y_j)$ a kernel function. The detailed proof can be found in Appendix C.

According to the specific form of the label kernel, we rewrite the objective function $\widehat{\text{EMMD}}^2\left(\hat{\mathcal{D}}^s_{\boldsymbol{W}}, \hat{\mathcal{D}}^t_{\boldsymbol{W}}\right)$ as $f(\boldsymbol{W})$,

$$\begin{aligned} f(\boldsymbol{W}) = & \sum_{i,j=1}^{n_s} h_{ij} k\left(\boldsymbol{W}^\top \boldsymbol{x}_i^s, \boldsymbol{W}^\top \boldsymbol{x}_j^s\right) + \sum_{i,j=1}^{n_t} h_{ij} k\left(\boldsymbol{W}^\top \boldsymbol{x}_i^t, \boldsymbol{W}^\top \boldsymbol{x}_j^t\right) \\ & -2\sum_{i,j=1}^{n_s,n_t} h_{ij} k\left(\boldsymbol{W}^\top \boldsymbol{x}_i^s, \boldsymbol{W}^\top \boldsymbol{x}_j^t\right), \end{aligned} \tag{14}$$

where h_{ij} is the (i,j)th element of the matrix $\boldsymbol{H}$. $\boldsymbol{H}$ is the Hadamard product (element-wise product) of $\boldsymbol{G}$ and $\boldsymbol{L}$. $\boldsymbol{G}$ is the Gram matrix of the label kernel $\tau(\cdot,\cdot)$ for all the source and target labels, and $\boldsymbol{L} = \boldsymbol{e}\boldsymbol{e}^\top$, where

$$\boldsymbol{e} = \left[\frac{1}{n_s} \ \cdots \ \frac{1}{n_s} \ -\frac{1}{n_t} \ \cdots \ -\frac{1}{n_t}\right]^\top \in \mathbb{R}^{(n_s+n_t)\times 1}. \tag{15}$$

Therefore, the final optimization problem is given as

$$\begin{aligned} & \min_{\boldsymbol{W}} f(\boldsymbol{W}) \\ & \text{s.t. } \boldsymbol{W}^\top \boldsymbol{W} = \mathbf{I}. \end{aligned} \tag{16}$$

2.4. Learning the mapping

In this section, we solve optimization problem (16) to learn the projection matrix $\boldsymbol{W}$. According to [13], this constrained problem can be regarded as an unconstrained problem defined on the Grassmann manifold $\text{Gr}(d, D)$. Hence, we can use the first-order Riemannian Gradient Descent (RGD) algorithm [1] to solve it.

In general, the RGD algorithm mainly consists of iteratively computing the Riemannian gradient of the objective function and performing a retraction to make sure that the solution stays on the manifold. On the Grassmann manifold $\text{Gr}(d, D)$, the Riemannian gradient with respect to the objective function $f(\boldsymbol{W})$ is expressed as

$$\nabla_{\boldsymbol{W}}(f) = \partial f_{\boldsymbol{W}} - \boldsymbol{W}\boldsymbol{W}^\top \partial f_{\boldsymbol{W}}, \tag{17}$$

where $\partial f_{\boldsymbol{W}}$ is the Euclidean gradient of f at $\boldsymbol{W}$. Besides, the retraction is given by

$$r_{\boldsymbol{W}}(\boldsymbol{Z}) = \boldsymbol{U}\boldsymbol{V}^\top, \tag{18}$$

where $(\boldsymbol{W} + \boldsymbol{Z}) = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ is the Singular Value Decomposition (SVD). To perform these two steps, we need to calculate the Euclidean gradient $\partial f_{\boldsymbol{W}}$ beforehand. For $f(\boldsymbol{W})$ in Eq. (14), its Euclidean gradient can be expressed as

$$\frac{\partial f}{\partial \boldsymbol{W}} = \sum_{i,j=1}^{n_s} h_{ij} \frac{\partial k\left(\boldsymbol{W}^\top \boldsymbol{x}_i^s, \boldsymbol{W}^\top \boldsymbol{x}_j^s\right)}{\partial \boldsymbol{W}} + \sum_{i,j=1}^{n_t} h_{ij} \frac{\partial k\left(\boldsymbol{W}^\top \boldsymbol{x}_i^t, \boldsymbol{W}^\top \boldsymbol{x}_j^t\right)}{\partial \boldsymbol{W}} - 2\sum_{i,j=1}^{n_s,n_t} h_{ij} \frac{\partial k\left(\boldsymbol{W}^\top \boldsymbol{x}_i^s, \boldsymbol{W}^\top \boldsymbol{x}_j^t\right)}{\partial \boldsymbol{W}}. \tag{19}$$

Apparently, this gradient is dependent on the choice of the feature kernel $k(\boldsymbol{W}^\top \boldsymbol{x}_i, \boldsymbol{W}^\top \boldsymbol{x}_j)$. For the linear, polynomial, and Gaussian kernels, their gradients with respect to $\boldsymbol{W}$ can be respectively expressed as follows: • Linear Kernel:

$$\frac{\partial k}{\partial \boldsymbol{W}} = \left(\boldsymbol{x}_i \boldsymbol{x}_j^\top + \boldsymbol{x}_j \boldsymbol{x}_i^\top\right)\boldsymbol{W}, \tag{20}$$

- Polynomial Kernel:

$$\frac{\partial k}{\partial \boldsymbol{W}} = deg(\boldsymbol{x}_i^\top \boldsymbol{W}\boldsymbol{W}^\top \boldsymbol{x}_j + 1)^{deg-1}\left(\boldsymbol{x}_i \boldsymbol{x}_j^\top + \boldsymbol{x}_j \boldsymbol{x}_i^\top\right)\boldsymbol{W}, \tag{21}$$

- Gaussian Kernel:

$$\frac{\partial k}{\partial \boldsymbol{W}} = -\frac{2}{\gamma} k(\boldsymbol{W}^\top \boldsymbol{x}_i, \boldsymbol{W}^\top \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{W}. \tag{22}$$

With the Euclidean gradient, we are ready to solve optimization problem (16) to obtain the feature projection matrix $\boldsymbol{W}$. Since the initial target labels estimated by a naive SVM classifier may be relatively inaccurate due to the joint distribution mismatch, we therefore constantly update the target labels. In particular, every time after we obtain the projection matrix, we use it to project the source and target data, and then update the labels for the projected target data by an SVM trained on the projected source data. In fact, iteratively optimizing the projection matrix and updating the target labels can promote the learning of each other. On the one hand, better target labels can be acquired from a better projection matrix, because the source and target joint distribution mismatch is lessened after the feature projection. On the other hand, better projection matrix can in turn be obtained from the more accurate target labels, since these labels together with the unlabeled target data better reflect the true target joint distribution. Note that the validity of this iterative strategy is empirically verified by a string of works that are based on marginal and class-conditional distribution matching [10,23,28,31,32,34,37,57].

The algorithm of JDME is summarized in Algorithm 1. In the algorithm, line 4 to line 8 are the RGD procedure to learn the projection matrix $\boldsymbol{W}$, which takes a computational complexity of $O\left(Dd^2 + d^3\right)$. Besides, the SVM classification takes $O(n_s^2)$. Suppose the maximum iteration for optimizing the projection matrix is fixed to t, and the maximum iteration for updating the target labels is fixed to T, then the total computational complexity is no more than $O\left(Tt\left(Dd^2 + d^3\right) + Tn_s^2\right)$.

Algorithm 1: Joint Distribution Matching Embedding

```
Input: the labeled source data $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
the unlabeled target data $\mathcal{D}^u = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$
the dimension $d$ of the latent space
Output: the target labels $\{\hat{y}_i^t\}_{i=1}^{n_t}$
1: Initialize the target labels $\{y_i^t\}_{i=1}^{n_t}$ using an SVM classifier trained on $\mathcal{D}^s$
2: Initialize $\boldsymbol{W}$ to the PCA projection matrix of $\boldsymbol{X} = [\boldsymbol{X}^s, \boldsymbol{X}^t]$
3: repeat
4:   repeat
5:     Compute the Riemannian gradient of $f(\boldsymbol{W})$ at $\boldsymbol{W}$ as $\nabla_{\boldsymbol{W}}(f) = \partial f_{\boldsymbol{W}} - \boldsymbol{W}\boldsymbol{W}^\top \partial f_{\boldsymbol{W}}$
6:     Update the solution as $\boldsymbol{W}_{new} \leftarrow \boldsymbol{U}\boldsymbol{V}^\top$, where $\boldsymbol{U}, \boldsymbol{V}^\top$ are obtained from the SVD decomposition of $(\boldsymbol{W} - \alpha\nabla_{\boldsymbol{W}}(f))$
7:     $\boldsymbol{W} \leftarrow \boldsymbol{W}_{new}$
8:   until convergence or reaching the maximum iteration
9:   Update the target labels $\{\hat{y}_i^t\}_{i=1}^{n_t}$ for $\{\boldsymbol{W}^\top \boldsymbol{x}_i^t\}_{i=1}^{n_t}$ by an SVM classifier trained on $\mathcal{D}_{\boldsymbol{W}}^s = \{(\boldsymbol{W}^\top \boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$
10: until convergence or reaching the maximum iteration
11: return the target label $\{\hat{y}_i^t\}_{i=1}^{n_t}$
```

2.5. Discussion

There are a number of works [23,28,31,32,34,37,52–54,57] jointly aligning the marginal distributions and class-conditional distributions for domain adaptation, which are highly-related to JDME. Therefore, here we make a comparison between JDME and these works. On the one hand, JDA [37] is a special case of JDME. To be specific, it can be observed that JDME reduces to JDA when the label kernel function is chosen as an all-one function plus a class-conditional function, where the all-one function is defined as $o(y_i, y_j) = 1,\ \forall y_i, y_j$. For other methods jointly matching the marginal and class-conditional distributions, to our knowledge, they consider the data structure [31,32], the class imbalance problem [53], the different importance of these two distributions [52,54], or other aspects to improve the performance of JDA. On the other hand, most of the works jointly aligning these two distributions compare the distributions in the original feature space. In their kernel extensions, the projection matrix is placed outside the kernel function. By contrast, we compare the distributions in the RKHS and place the projection matrix inside the kernel function in JDME. Under such circumstances, as analyzed in [27], if the feature kernel function k is the polynomial kernel of degree deg, the kernel embedding tries to match the distributions up to the degth moment. And if the kernel function k is the Gaussian kernel, the kernel embedding tries to match all moments of the source and target distributions. This statistical property helps JDME to align the distributions better.

3. Experiments

In this section, we comprehensively evaluate the performance of JDME on a series of real-world image and text datasets. In particular, we first introduce the datasets, then describe the experimental settings, report the results with statistical test, and finally perform empirical analysis of the proposed method.

3.1. Datasets

We show the information of all the datasets in Table 1. Samples of these image datasets are shown in Figs. 1, 2, 3 and 4. Each dataset is further described as follows.

Office31 [46] is a popular object recognition dataset which consists of 3 domains: Amazon (A), DSLR (D), and Webcam (W). Each domain has 31 categories. Following [39], we use two kinds of deep features for this dataset: the FC7 activations from the VGG-M [6]

Table 1
Information of the Domain Adaptation Datasets.

Dataset	Feature(size)	#Classes	Domain (Abbreviation)	#Examples
Office31	VGG-M(4096)	31	Amazon (A)	2871
	VGG-VD-16(4096)		DSLR (D)	498
	ResNet-50(2048)		Webcam (W)	795
Office-Caltech	VGG-M(4096) VGG-VD-16(4096)	10	Amazon (A)	958
			Caltech (C)	1123
			DSLR (D)	157
			Webcam (W)	295
COIL	gray-scale(1024)	20	COIL1 (C1)	720
			COIL2 (C2)	720
SPAM	bag-of-words(4272)	2	Enron (E)	4205
			SMS (S)	5338
PIE	gray-scale(1024)	68	looking forward (C27)	1407
			looking down (C09)	1407
			looking left(α_1)[a] (C05)	1407
			looking left(α_2)a (C37)	1407
			looking left(α_3)a (C25)	1407
			looking left(α_4)a (C02)	1407
Office-Home	ResNet-50(2048)	65	Art (A)	2427
			Clipart (C)	4365
			Product (P)	4439
			RealWorld (R)	4357

[a] α_i means the degree of looking left, where $\alpha_1 < \alpha_2 < \alpha_3 < \alpha_4$.

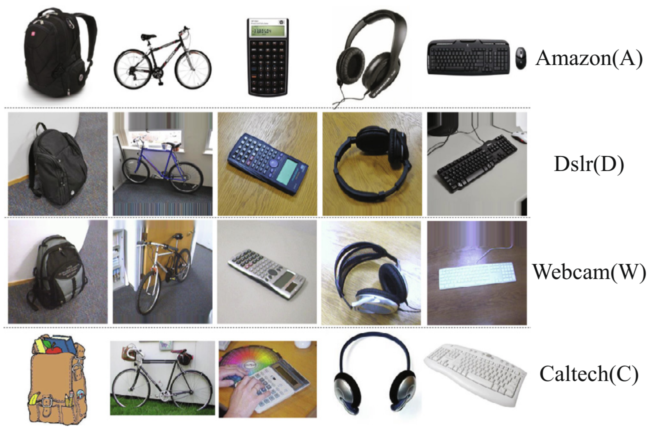


Fig. 1. Samples from Office-Caltech.

Fig. 3. Samples from PIE.

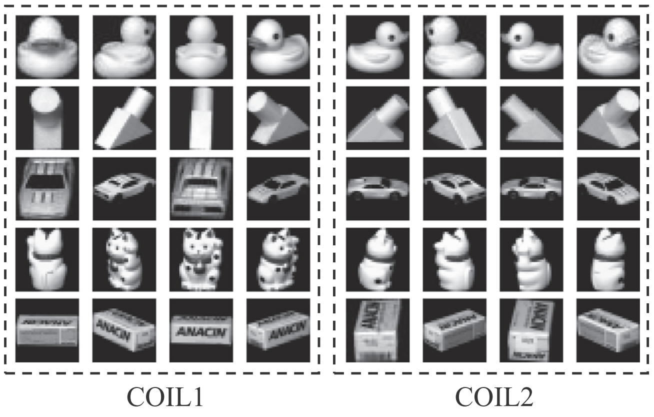


Fig. 2. Samples from COIL.

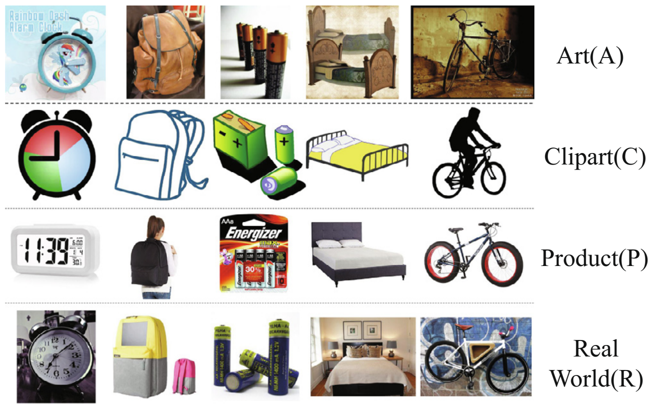


Fig. 4. Samples from Office-Home.

model and the FC7 activations from the VGG-VD-16 [48] model to contrast JDME with shallow methods. For fare comparison, we extract a feature vector from the ResNet-50 model to compare JDME with deep learning methods. Since there are 3 domains, we construct 6 domain adaptation tasks by training on one domain and testing on another.

Office-Caltech [17] includes 4 domains: Amazon (A), Caltech (C), DSLR (D) and Webcam (W). Each domain has 10 categories. We use the FC7 activations from the VGG-M [6] model and the FC7 activations from the VGG-VD-16 [48] model as two kinds of features and build 12 transfer tasks in total.

COIL [42] is another object recognition dataset that contains 1440 samples over 20 classes with the image size 32×32. We split the dataset into two subsets COIL1 (C1) and COIL2 (C2) according to the capture directions [32,37]. Specifically, COIL1 contains all the images captured in the directions of $[0^\circ, 85^\circ] \cup [180^\circ, 265^\circ]$ while COIL2 contains the remaining directions. We construct two transfer tasks from COIL: C1 $\rightarrow$ C2 and C2 $\rightarrow$ C1.

SPAM [26] is a text dataset that includes two subsets from the UCI machine learning repository: the Enron (E) spam database [25] and the SMS-spam (S) dataset [2]. The Enron spam dataset contains 4205 emails, while the SMS-spam dataset contains 5338 text messages. The task is to detect whether a given email or message is spam. We use the 4272-dimensional bag-of-words feature vector extracted by [26], and construct 2 domain adaptation tasks from this dataset.

PIE [47] is a face recognition dataset made up of 41,368 images of 68 people taken from different views, poses, illumination and expressions. We choose the same subsets as [22], namely looking-forward (C27), looking-downward (C09), looking towards left in an increasing angle (C05, C37, C25, C02). Each gray-scale image is cropped into size 32×32 and then flattened into a 1024-dimensional feature vector. We construct 30 domain adaptation tasks by choosing any two subsets as a pair of source domain and target domain.

Office-Home [51] is a large image dataset which consists of 4 subsets: Art (artistic depictions of objects), Clipart (clipart images), Product (images of objects without a background), Real World (images of objects captured with a regular camera). These domains are respectively abbreviated as A, C, P and R in the experiment. We extract a 2048-dimensional feature vector using the ResNet-50 [21] model for this dataset, and build 12 domain adaptation tasks.

3.2. Experiment setup

We denote our JDME method with the linear feature kernel as JDME-L, with the polynomial kernel as JDME-P, and with Gaussian kernel as JDME-G. In particular, the degree of the polynomial kernel is set to be 2, i.e. $deg = 2$, and the scale parameter γ of the Gaussian kernel is set to be the median square distance between the source and target samples. We use the pymanopt [50] package to perform the RGD algorithm, and fix the maximum iteration to $t = 5$. Empirically, we observe that this number is sufficient for the objective to converge. Besides, the maximum iteration for updating the target labels is fixed to $T = 10$ for deeper features and $T = 30$ for rawer features.

Table 2
Classification accuracy (%) of different algorithms on 40 cross-domain transfer tasks.

Datasets	Tasks	SVM	SA [14]	CORAL [49]	DME-MMD [3]	LDADA [39]	OT [11]	KWC [59]	KOT [59]	RT [27]	DICE [32]	LPJT [28]	JDME-L	JDME-P	JDME-G
Office31 VGG-M	A→D	64.61	59.84	60.24	65.41	**75.15**	60.83	68.59	67.00	59.84	67.79	73.36	72.56	71.97	*72.76*
	A→W	60.25	56.35	59.87	63.52	66.67	63.52	63.90	63.65	58.74	65.03	69.94	68.55	**72.20**	*69.69*
	D→A	52.22	55.24	52.47	56.87	59.35	60.42	43.10	55.34	53.39	62.19	56.59	68.44	*68.34*	**68.55**
	D→W	91.57	**96.60**	94.97	91.32	85.28	88.81	86.29	91.07	*95.22*	96.23	91.19	92.70	93.84	92.45
	W→A	53.71	54.56	52.72	56.16	66.13	62.30	54.35	56.76	53.18	60.84	60.53	*67.91*	**68.34**	68.02
	W→D	98.41	98.01	98.01	97.81	93.64	91.45	91.65	95.83	97.81	98.41	97.61	98.41	**98.61**	98.41
	Average	70.13	70.10	69.71	71.85	74.37	71.22	67.98	71.61	69.70	75.08	74.87	*78.10*	**78.88**	78.31
Office31 VGG-VD-16	A→D	71.57	63.22	68.39	73.76	76.34	76.54	75.55	74.35	69.18	77.53	79.72	80.91	**81.51**	80.91
	A→W	68.18	62.39	67.67	71.32	76.73	76.73	74.84	73.84	66.79	77.11	77.74	*83.14*	**84.65**	83.77
	D→A	57.54	61.38	58.36	62.58	68.34	59.18	49.49	61.02	58.79	66.95	61.13	*70.15*	**70.82**	70.18
	D→W	95.22	95.85	95.72	95.22	91.57	94.21	91.32	93.33	96.23	94.72	94.72	96.98	**97.36**	96.98
	W→A	57.15	59.46	57.90	60.84	67.52	60.53	57.26	61.24	57.97	67.41	64.79	68.90	**69.47**	*68.69*
	W→D	*98.41*	*98.41*	98.21	98.21	96.42	97.02	95.83	98.01	98.61	96.22	97.81	*98.41*	**98.81**	98.21
	Average	74.68	73.45	74.38	76.99	79.49	77.37	74.05	76.97	74.59	79.99	79.32	*83.08*	**83.77**	83.13
Office-Caltech VGG-M	A→C	86.38	77.56	83.70	85.40	88.25	87.98	87.89	88.16	83.62	87.89	**88.51**	88.33	88.16	88.33
	A→D	87.90	73.89	81.53	88.54	89.81	**92.36**	89.17	91.08	79.62	89.17	**92.36**	*91.72*	*91.72*	91.08
	A→W	83.05	76.61	74.58	87.80	92.54	91.53	90.51	90.85	76.95	89.49	91.53	92.88	92.88	92.88
	C→A	93.32	89.98	89.14	93.01	**94.99**	93.95	93.63	93.42	89.67	94.47	94.15	*94.36*	93.84	93.95
	C→D	91.72	87.90	83.44	91.08	91.08	91.08	91.08	89.17	79.62	92.99	87.90	**98.73**	*98.09*	**98.73**
	C→W	83.73	83.05	79.66	89.49	*94.24*	93.22	92.20	91.19	75.93	93.22	93.56	*94.24*	**95.59**	94.92
	D→A	88.83	88.62	84.55	90.19	94.15	91.86	81.00	89.67	85.80	94.26	82.05	94.68	*94.47*	**94.78**
	D→C	81.21	81.48	78.90	82.55	84.68	82.90	72.66	81.75	79.34	85.75	77.74	**88.42**	*85.84*	**88.42**
	D→W	97.63	97.29	99.32	*99.66*	95.25	98.64	93.56	*99.66*	99.32	97.29	99.32	*99.66*	**100.00**	**100.00**
	W→A	87.27	89.14	82.05	91.44	94.05	93.42	92.48	91.86	82.67	94.05	92.59	93.84	94.05	**94.26**
	W→C	81.83	81.66	74.80	82.81	**88.25**	86.38	85.13	85.66	75.33	87.53	87.53	86.20	85.31	86.29
	W→D	*98.73*	*98.73*	**100.00**	*98.73*	*98.73*	**100.00**	*98.73*	97.45	*98.73*	*98.73*	*98.73*	*98.73*	*98.73*	*98.73*
	Average	88.47	85.49	84.31	90.06	92.17	91.94	89.00	90.83	83.88	92.07	90.50	93.48	*93.22*	**93.53**
Office-Caltech VGG-VD-16	A→C	90.38	82.55	86.11	89.40	*91.63*	89.40	89.58	90.03	86.02	90.12	90.65	91.99	91.36	**92.34**
	A→D	89.81	81.53	78.34	89.17	**98.09**	89.81	89.17	87.90	71.97	91.72	90.45	92.36	92.36	92.36
	A→W	87.46	80.34	76.95	87.46	**96.61**	**96.61**	95.25	95.59	77.63	94.92	95.59	96.27	95.59	**96.61**
	C→A	94.78	91.75	91.96	94.68	**95.82**	94.15	*95.09*	94.68	89.46	94.78	94.99	94.78	94.89	95.09
	C→D	89.17	81.53	78.98	92.36	96.82	92.36	95.54	92.36	79.62	91.72	91.72	**97.45**	*95.54*	*95.54*
	C→W	88.14	86.78	82.37	93.22	*97.29*	*97.29*	94.58	95.59	80.68	**97.63**	**97.63**	96.95	*97.29*	96.95
	D→A	87.79	85.39	84.34	88.31	94.99	89.35	82.05	88.73	84.24	93.53	87.58	**96.03**	*95.51*	**96.03**
	D→C	84.68	80.32	80.94	83.97	**91.36**	83.44	74.98	80.85	81.66	87.98	80.23	87.53	88.16	*88.07*
	D→W	99.32	99.32	98.64	98.31	98.64	97.63	95.25	97.63	99.32	98.64	98.31	99.66	**100.00**	99.66
	W→A	89.46	89.25	84.03	90.81	95.82	94.47	93.22	93.01	86.22	95.09	95.30	95.72	95.82	**95.93**
	W→C	87.18	84.95	83.26	88.25	**92.25**	89.67	87.98	89.05	83.79	90.38	90.29	*90.56*	89.49	90.83
	W→D	**99.36**	**99.36**	**99.36**	**99.36**	**99.36**	**99.36**	98.73	98.73	**99.36**	**99.36**	98.73	**99.36**	**99.36**	**99.36**
	Average	90.63	86.92	85.44	91.27	**95.72**	92.80	90.95	92.01	85.00	93.82	92.62	*94.89*	94.61	94.90
COIL gray-scale	C1→C2	79.03	81.25	81.53	86.53	77.92	86.94	89.03	*91.11*	86.53	**92.50**	88.06	87.78	91.39	88.89
	C2→C1	76.53	83.61	83.89	85.69	81.53	87.08	88.33	*90.42*	90.00	**94.40**	87.78	87.78	92.92	87.36
	Average	77.78	82.43	82.71	86.11	79.72	87.01	88.68	*90.765*	88.26	**93.45**	87.92	87.78	92.15	88.13
SPAM bag-of-words	E → S	30.09	*65.16*	53.02	46.97	67.16	49.40	46.03	46.10	54.31	46.70	50.24	64.63	63.64	**69.91**
	S → E	55.05	55.03	52.72	52.84	46.11	53.53	50.68	51.56	50.49	*57.27*	49.18	**59.48**	58.60	56.53
	Average	42.57	60.09	52.87	49.90	56.64	51.47	48.35	48.83	52.40	51.98	49.71	62.05	*61.12*	**63.22**
Average Rank		9.56	10.24	10.95	8.65	5.67	7.30	10.54	9.04	10.63	5.83	6.90	*3.51*	3.13	**3.06**

1.The best result of each task is highlighted in **bold**, the second best is underlined and the third best is marked in *italic*.
2.The average rank of each method is calculated over all the tasks, not including the average accuracy on the datasets.

We compare our method with state-of-the-art shallow approaches, including Subspace Alignment (SA) [14], Correlation Alignment (CORAL) [49], Distribution-Matching Embedding with the MMD (DME-MMD) [3], LDA-inspired Domain Adaptation (LDADA) [39], Optimal Transport (OT) [11], Kernel Optimal Transport (KOT) [59], Kernel Whitening-Coloring (KWC) [59], Domain-Irrelevant Class Clustering (DICE) [32], Locality Preserving Joint Transfer (LPJT) [28], and Residual Transformation (RT) [27]. These comparison methods are implemented by using the source codes provided by the authors if they are available, or by following the algorithms proposed in the corresponding papers. For classification, we use a linear SVM [5] for all the methods. Almost all the methods have hyper-parameters, following [28], we tune the hyper-parameters for each method and then report its best result. Specifically, for our JDME-L, JDME-P and JDME-G, they all include a common hyper-parameter, the dimension d of the latent space, which is searched from the set $\{30, 50, 100, 200, 350, 500\}$.

Besides, we also compare our approach with state-of-the-art deep learning methods on the Office-Home and the Office31 datasets. The features are both extracted from the ResNet-50 model. These deep learning methods include Deep Adaptation Network (DAN) [35], Domain-Adversarial Neural Network (DANN) [15], Joint Adaptation Network (JAN) [38], The Entropy conditioning variant of Conditional Domain Adversarial Network (CDAN + E) [36], and Batch Spectral Penalization (BSP + CDAN) [9]. Note that the results of these deep domain adaptation methods are directly cited from [9].

3.3. Experiment results

Here we first report the experiment results. Following that, we conduct a statistical test to check whether the proposed method is superior to its competitors.

3.3.1. Presentation of the results

Table 2 and Fig. 5 report the classification results of the domain adaptation methods on cross-domain transfer tasks constructed from the image and text datasets. On each transfer task in Table 2, the best result is highlighted in **bold**, the second-best is underlined, and the third-best is marked in *italic*. Moreover, to summarize the performance of the methods, following [30], we also compute the average rank for each one of them over all the tasks. Briefly speaking, the smaller the rank is, the better the method performances. From Table 2, we observe that JDME-L, JDME-P, and JDME-G obtain the three smallest average ranks 3.51, 3.13 and 3.06, which indicates that their overall performances are better than the other methods. Similarly, Fig. 5 shows that the proposed method achieves higher accuracy in most of the tasks built from the PIE dataset, and JDME-G reaches the best average accuracy. The comparison results with the deep learning methods are reported in Table 3 and Fig. 6. It can be observed from Table 3 that although our JDME with three kinds of kernels does not significantly outperform the deep method BSP + CDAN, it yields comparable performance to it. In fact, our JDME-G obtains an average classification accuracy of 65.95%, which is very close to the best Result 66.30% produced by BSP + CDAN. Fig. 6 likewise reports JDME can achieve a comparable performance to these deep learning methods. These results verify that our JDME strategy based on the EMMD metric is effective for the UDA problem, and that even in such a simple form it still has its advantage over the existing shallow domain adaptation methods.

3.3.2. Statistical test

Based on Table 2, we further conduct the Wilcoxon signed-ranks test [8,12,56] to check whether the proposed method is significantly better than the other shallow methods. The null hypothesis is that the two algorithms a and b work equally well over multiple tasks. We rank all the tasks according to the absolute value of the accuracy difference between the two algorithms from small to large. The accuracy of algorithm a, as the minuend, is fixed as the accuracy of JDME-P. The rank rules are the same as the rules in Section 3.3.1. For each algorithm b, the statistic is computed as:

$$T(a,b) = \min\{R^+(a,b), R^-(a,b)\}, \tag{23}$$

where $R^+(a,b)$ means the sum of the ranks for the tasks on which algorithm a outperforms algorithm b, and $R^-(a,b)$ means the sum of the ranks for the opposite. As to the tasks on which the two algorithms work equally well, the ranks are averagely assigned to $R^+(a,b)$ and $R^-(a,b)$. Under these definitions, $R^+(a,b)$ and $R^-(a,b)$ are calculated as

$$R^+(a,b) = \sum_{a_i > b_i} \mathrm{rank}(|d_i|) + \frac{1}{2}\sum_{a_i = b_i} \mathrm{rank}(|d_i|), \tag{24}$$

$$R^-(a,b) = \sum_{a_i < b_i} \mathrm{rank}(|d_i|) + \frac{1}{2}\sum_{a_i = b_i} \mathrm{rank}(|d_i|), \tag{25}$$

where a_i and b_i mean the accuracy on the ith task for algorithm a and b, d_i means the accuracy discrepancy between algorithm a and b, computed as $d_i = a_i - b_i$.

The statistic of the Wilcoxon signed-ranks test is:

$$z(a,b) = \frac{T(a,b) - m(m+1)/4}{\sqrt{m(m+1)(2m+1)/24}}, \tag{26}$$

where m is the number of the tasks.

We use the accuracy from Table 2 to compute $z(a,b)$. As reported in Table 4, the values of other domain adaptation methods are all below -1.96. This indicates that JDME-P is statistically better than its competitors with a significance level $\alpha = 0.05$. For the z values of JDME-L and JDME-G, these values are greater than -1.96, which indicates that JDME-P does not statistically outperform JDME-L and JDME-G.

3.4. Empirical analysis

3.4.1. Justification of the empirical EMMD

Here we experimentally justify the empirical EMMD in Eq. (5), particularly in the use of the product kernel for the feature $\boldsymbol{x}$ and the label y. The goal is to show that in a supervised classification problem where the label y is dependent on its feature $\boldsymbol{x}$, our empirical EMMD metric $\widehat{\mathrm{EMMD}}(\mathcal{D}^s, \mathcal{D}^t)$ can produce reasonable estimate of the distance between the joint distributions that generate the samples $\mathcal{D}^s$ and $\mathcal{D}^t$. To this end, we construct two cases to test the validity of this metric: 1) the source and target joint distributions are exactly the same, and 2) the source and target joint distributions are vastly different. For simplicity, we consider a binary classification problem where the label $y \in \{-1, 1\}$.

We first construct two joint distributions for experiments. Note that a joint distribution $P_{XY}(\boldsymbol{x}, y)$ can be decomposed into $P_{XY}(\boldsymbol{x}, y) = P_X(\boldsymbol{x})P_{Y|X}(y|\boldsymbol{x})$. Therefore, $P_{XY}(\boldsymbol{x}, y)$ can be defined via defining its components $P_X(\boldsymbol{x})$ and $P_{Y|X}(y|\boldsymbol{x})$. Specifically, for the conditional distribution $P_{Y|X}(y|\boldsymbol{x})$, we only need to define $P_{Y|X}(y = 1|\boldsymbol{x})$ since $P_{Y|X}(y = -1|\boldsymbol{x}) = 1 - P_{Y|X}(y = 1|\boldsymbol{x})$. Now let the first joint distribution be $P^1_{XY}(\boldsymbol{x}, y) = P^1_X(\boldsymbol{x})P^1_{Y|X}(y|\boldsymbol{x})$. $P^1_X(\boldsymbol{x})$ and $P^1_{Y|X}(y|\boldsymbol{x})$ are defined as

$$P^1_X(\boldsymbol{x}) = \frac{1}{2}\mathcal{N}\left(\boldsymbol{x}, \mu^1_{-1}, \Sigma^1_{-1}\right) + \frac{1}{2}\mathcal{N}\left(\boldsymbol{x}, \mu^1_1, \Sigma^1_1\right), \tag{27}$$

$$P^1_{Y|X}(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(x_1 - x_2)}, \tag{28}$$

where $\boldsymbol{x} = (x_1, x_2)^\top$, and $\mathcal{N}(\boldsymbol{x}, \mu, \Sigma)$ is a Gaussian distribution with mean vector μ and covariance matrix Σ. The distribution parameters in Eq. (27) are defined as

$$\mu^1_{-1} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}, \ \mu^1_1 = \begin{bmatrix} 2 \\ 5 \end{bmatrix}, \ \Sigma^1_{-1} = \Sigma^1_1 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

Additionally, let the second joint distribution be $P^2_{XY}(\boldsymbol{x}, y) = P^2_X(\boldsymbol{x})P^2_{Y|X}(y|\boldsymbol{x})$, where $P^2_X(\boldsymbol{x})$ and $P^2_{Y|X}(y|\boldsymbol{x})$ are respectively defined as

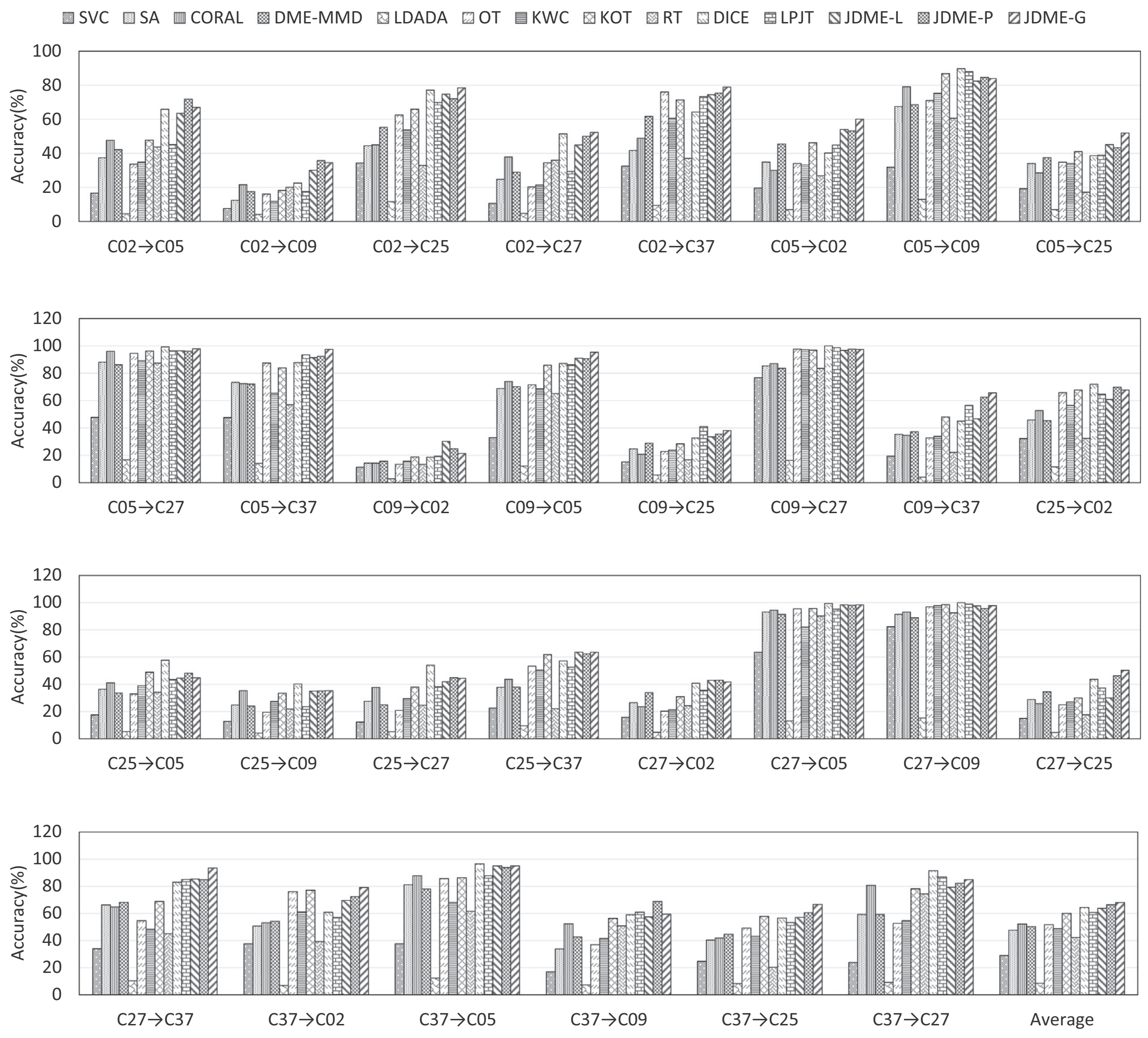


Fig. 5. Classification accuracy (%) of different methods on the PIE dataset. The average accuracy (%) of these methods over the 30 transfer tasks is 29.06 (SVC), 47.70 (SA), 52.19 (CORAL), 50.43 (DME-MMD), 8.70 (LDADA), 51.81 (OT), 48.90 (KWC), 59.99 (KOT), 42.38 (RT), *64.43* (DICE), 60.64 (LPJT), 63.80 (JDME-L), 66.43 (JDME-P), and **68.12** (JDME-G).

Table 3
Classification accuracy (%) of different algorithms on the Office-Home dataset with the ResNet-50 features.

Tasks	ResNet-50 [21]	DAN [35]	DANN [15]	JAN [38]	CDAN + E [36]	BSP + CDAN [9]	JDME-L	JDME-P	JDME-G
A→C	34.90	43.60	45.60	45.90	50.70	52.00	53.10	**53.31**	*52.94*
A→P	50.00	57.00	59.30	61.20	70.60	68.60	74.79	**74.90**	*74.61*
A→R	58.00	67.90	70.10	68.90	76.00	76.10	77.76	**77.87**	*77.62*
C→A	37.40	45.80	47.00	50.40	57.60	**58.00**	*56.57*	55.79	56.28
C→P	41.90	56.50	58.50	59.70	70.00	70.30	70.71	*70.60*	**70.85**
C→R	46.20	60.40	60.90	61.00	70.00	70.20	70.94	**71.06**	*70.90*
P→A	38.50	44.00	46.10	45.80	57.40	**58.60**	56.65	56.61	*56.98*
P→C	31.20	43.60	43.70	43.40	**50.90**	50.20	47.84	47.72	*48.36*
P→R	60.40	67.70	68.50	70.30	77.30	77.60	78.47	*78.40*	**78.6**1
R→A	53.90	63.10	63.20	63.90	70.90	**72.20**	65.22	65.60	*66.75*
R→C	41.20	51.50	51.80	52.40	56.70	**59.30**	55.92	55.72	*55.92*
R→P	59.90	74.30	76.80	76.80	*81.60*	**81.90**	81.41	81.23	81.62
Average Accuracy	46.10	56.30	57.60	58.30	*65.80*	**66.30**	65.78	65.73	65.95
Average Rank	9.00	7.92	6.71	6.38	3.42	**2.67**	*3.00*	3.25	**2.67**

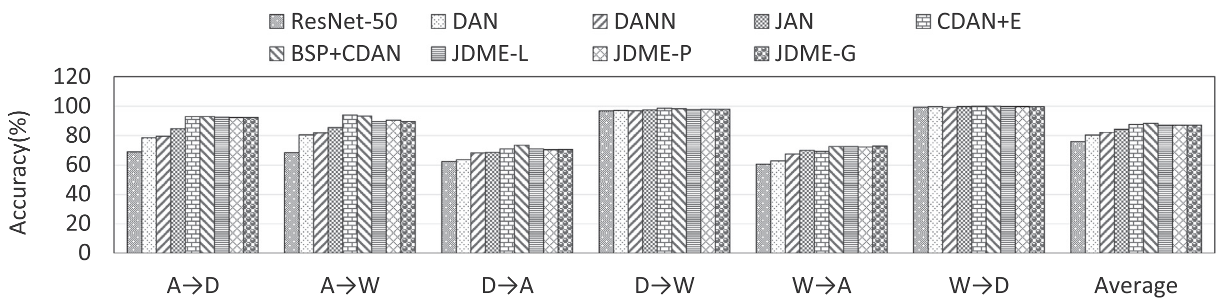


Fig. 6. Classification accuracy (%) of different methods on the Office31 dataset based on the ResNet-50 model. The average accuracy (%) of these methods over the 6 transfer tasks is 76.08 (ResNet-50), 80.37 (DAN), 82.22 (DANN), 84.32 (JAN), <u>87.65</u> (CDAN + E), **88.45** (BSP + CDAN), 87.21 (JDME-L), *87.22* (JDME-P), 87.14 (JDME-G).

Table 4
Statistics of the Wilcoxon signed-ranks test via fixing JDME-P as the control algorithm.

Methods	SVM	SA	CORAL	DME-MMD	LDADA
z	−5.82	−5.64	−5.71	−5.82	−3.26
Methods	OT	KWC	KOT	RT	DICE
z	−5.32	−5.78	−5.77	−5.73	−4.80
Methods	LPJT	JDME-L	JDME-G		
z	−4.94	−1.21	−1.69		

$$P_X^2(\boldsymbol{x}) = \frac{1}{2}\mathcal{N}\left(\boldsymbol{x}, \mu_{-1}^2, \Sigma_{-1}^2\right) + \frac{1}{2}\mathcal{N}\left(\boldsymbol{x}, \mu_1^2, \Sigma_1^2\right), \tag{29}$$

$$P_{Y|X}^2(y = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(x_2 - x_1)}. \tag{30}$$

The distribution parameters in Eq. (29) are defined as

$$\mu_{-1}^2 = \begin{bmatrix} 8 \\ 4 \end{bmatrix}, \ \mu_1^2 = \begin{bmatrix} 4 \\ 8 \end{bmatrix}, \ \Sigma_{-1}^2 = \Sigma_1^2 = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

In the first case, let the source and target joint distributions be $P_{XY}^s(\boldsymbol{x}, y) = P_{XY}^t(\boldsymbol{x}, y) = P_{XY}^1(\boldsymbol{x}, y)$. In the second case, let the source joint distribution be $P_{XY}^s(\boldsymbol{x}, y) = P_{XY}^1(\boldsymbol{x}, y)$, and the target joint distribution be $P_{XY}^t(\boldsymbol{x}, y) = P_{XY}^2(\boldsymbol{x}, y)$. Note that according to (28) and (30), the label y is always dependent on its feature $\boldsymbol{x}$. Fig. 7 plots the samples drawn from the source and target joint distributions in these two cases. For both cases, the estimate $\widehat{\text{EMMD}}(\mathcal{D}^s, \mathcal{D}^t)$ can be computed by drawing random samples $\mathcal{D}^s$ from $P_{XY}^s(\boldsymbol{x}, y)$, and $\mathcal{D}^t$ from $P_{XY}^t(\boldsymbol{x}, y)$. If the estimate is reasonable, then it should be a small value close to zero in the first case, and a large value far away from zero in the second case. To verify this point, we drawn equal number of source samples and target samples, i.e. $n_s = n_t$, and let the sample size vary in the range $\{100, 500, 1000, 1500, 2000, 3000\}$. For each size, we repeat the random sampling procedure for 10 times to calculate the mean and standard deviation of $\widehat{\text{EMMD}}(\mathcal{D}^s, \mathcal{D}^t)$. The results are plotted in Fig. 8. From Fig. 8(a) for the first case, we observe that $\widehat{\text{EMMD}}(\mathcal{D}^s, \mathcal{D}^t)$ is very close to zero as long as enough samples are provided. On the contrary, from Fig. 8(b) for the second case, we notice that $\widehat{\text{EMMD}}(\mathcal{D}^s, \mathcal{D}^t)$ becomes a pretty large value, which is reasonable since the source and target joint distributions are very different. These results, to a certain extent verify that our empirical metric in Eq. (5) is reasonable. More importantly, they also show that a product kernel works for the case where the feature $\boldsymbol{x}$ and the label y are dependent.

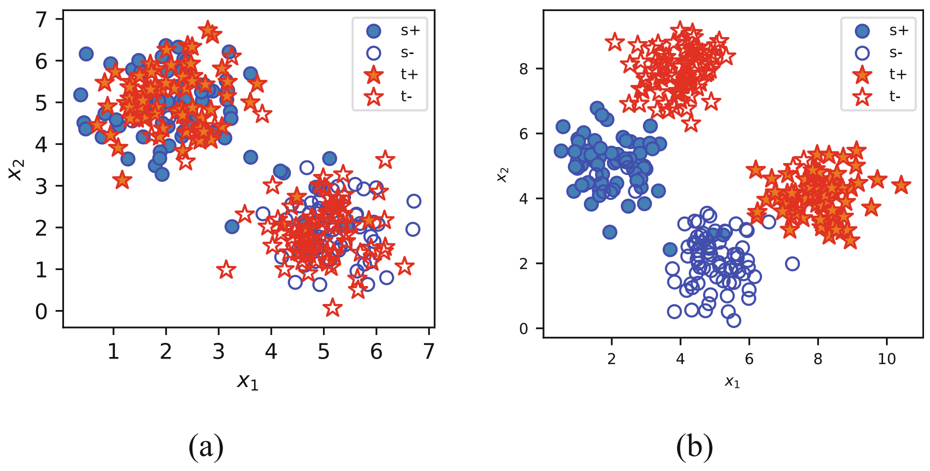


Fig. 7. Samples drawn from the source and target joint distributions, where s+, s-, t+, t- respectively represent the positive source sample, negative source sample, positive target sample, and negative target sample. (a) The source and target joint distributions are the exactly same: $P_{XY}^s(\boldsymbol{x}, y) = P_{XY}^t(\boldsymbol{x}, y) = P_{XY}^1(\boldsymbol{x}, y)$. (b) The source and target joint distributions are vastly different: $P_{XY}^s(\boldsymbol{x}, y) = P_{XY}^1(\boldsymbol{x}, y), P_{XY}^t(\boldsymbol{x}, y) = P_{XY}^2(\boldsymbol{x}, y)$.

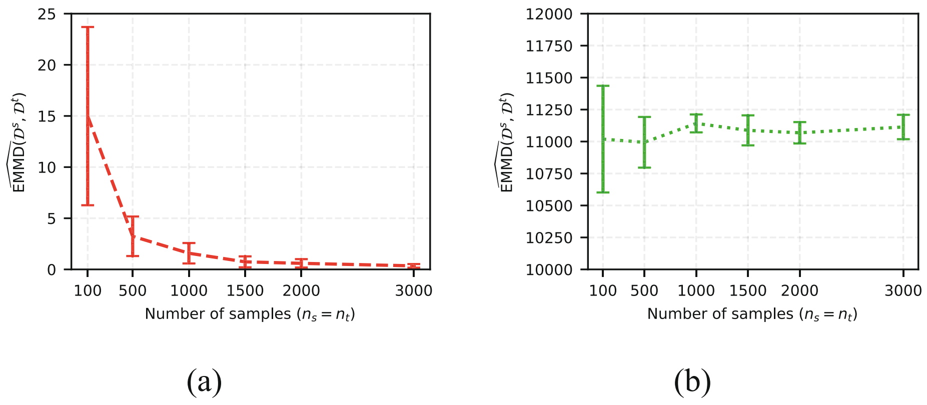


Fig. 8. Mean and standard deviation of $\widehat{\text{EMMD}}(\mathcal{D}^s, \mathcal{D}^t)$, where $\mathcal{D}^s$ is sampled from $P^s_{XY}(\boldsymbol{x}, y)$, and $\mathcal{D}^t$ from $P^t_{XY}(\boldsymbol{x}, y)$. (a) The source and target joint distributions are the exactly same: $P^s_{XY}(\boldsymbol{x}, y) = P^t_{XY}(\boldsymbol{x}, y) = P^1_{XY}(\boldsymbol{x}, y)$. (b) The source and target joint distributions are vastly different: $P^s_{XY}(\boldsymbol{x}, y) = P^1_{XY}(\boldsymbol{x}, y), P^t_{XY}(\boldsymbol{x}, y) = P^2_{XY}(\boldsymbol{x}, y)$.

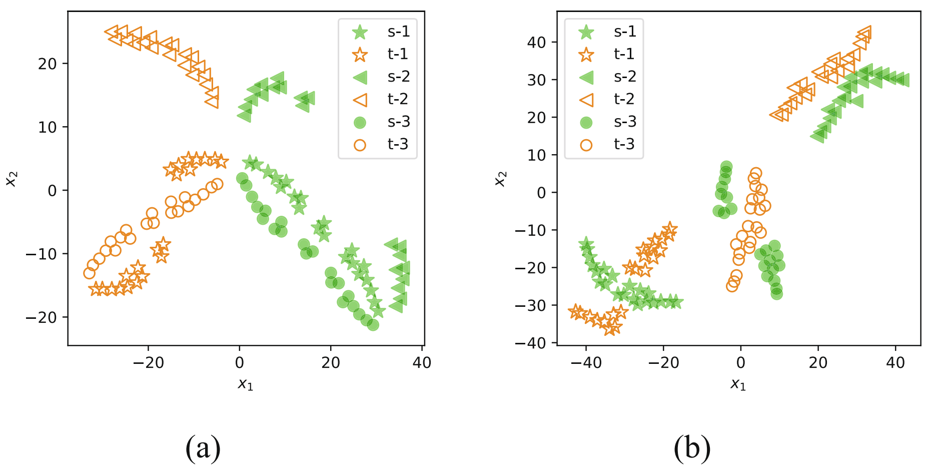


Fig. 9. The data distributions of C27 and C37 from the PIE dataset. The solid shapes are used to represent the source samples and the hollow shapes are used to represent the target samples. Different shapes are used to denote samples from different classes. (a) The original data distributions of C27 and C37. (b) The matched data distributions of C27 and C37 by the proposed JDME method.

3.4.2. Effect of joint distribution matching

In this part, we use t-SNE [40] to visualize the effect of joint distribution matching. Take JDME with polynomial kernel with degree 2 as an example. We conduct experiments on the PIE dataset and choose C27 and C37 as source and target domain, respectively. For the sake of succinctness, we take samples from the first three classes to show the data distribution difference before and after matching. As shown in Fig. 9, samples gather according to their domains before domain adaptation, while after matching, samples of the same class from different domains cluster together. This corresponds to the classification accuracy, which is 33.97% (SVM) before matching and 84.97% (JDME-P) after. The results imply the joint distribution matching can generate not only domain-invariant but class-discriminative features.

3.4.3. Convergence analysis

In this part, we analyze the convergence of JDME-P. We conduct experiments on C2 → C1 of COIL and A → W of Office31 based on the features from the VGG-VD-16 model to discuss the trend of objective value in Eq. (5) with the increasing of the iteration numbers. As is shown in Fig. 10, after about 10 iterations, the value of the objective function has become stable, showing our method tends to converge quickly.

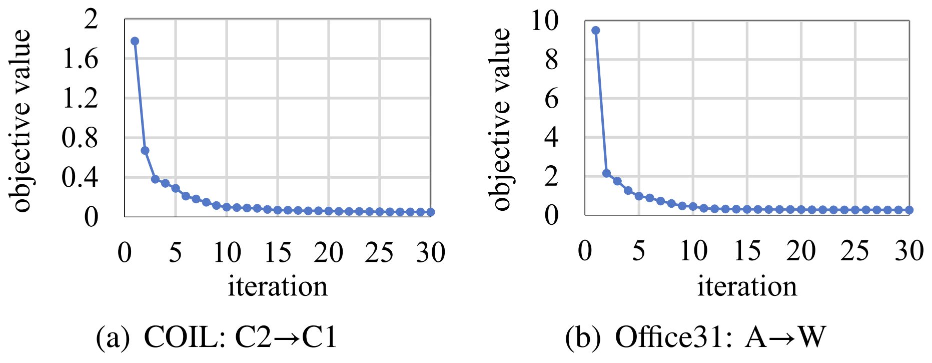


Fig. 10. Convergence curves of JDME-P on COIL: C2→C1 and Office31: A→W.

Table 5
Classification accuracy (%) of JDME-G with different dimensions on Office31 and Office-Caltech.

DataSets	Tasks	30	50	100	200	350	500
Office31 VGG-VD-16	A→D	80.12	79.13	79.92	81.11	80.91	80.91
	A→W	78.99	78.74	80.88	83.90	83.02	83.77
	D→A	67.02	69.29	68.12	68.48	68.44	70.18
	D→W	94.34	96.98	97.11	96.86	96.73	96.98
	W→A	67.80	67.98	68.12	68.37	68.83	68.69
	W→D	97.42	98.01	98.21	98.21	98.21	98.21
	Average	80.95	81.69	82.06	82.82	82.69	83.13
Office-Caltech VGG-VD-16	A→C	92.16	92.43	91.99	92.34	92.34	92.52
	A→D	92.36	92.36	92.36	92.36	92.36	92.36
	A→W	94.92	95.59	96.27	96.61	95.93	96.95
	C→A	93.84	95.09	94.78	95.09	95.09	95.20
	C→D	98.09	98.09	97.45	95.54	96.18	95.54
	C→W	96.27	96.61	96.95	96.95	96.95	96.95
	D→A	95.09	94.89	95.51	96.03	95.93	95.82
	D→C	88.60	88.60	88.16	88.07	88.07	88.16
	D→W	100.00	100.00	99.66	99.66	99.66	99.66
	W→A	95.62	95.62	95.62	95.93	95.72	95.93
	W→C	89.14	89.67	89.85	90.83	90.74	90.56
	W→D	98.73	99.36	99.36	99.36	99.36	99.36
	Average	94.57	94.86	94.83	94.90	94.86	94.92

3.4.4. Hyperparameter sensitivity

Our JDME approach only includes one hyperparameter, the dimension d of the subspace. Here we analyze the sensitivity of JDME-G with respect to this parameter. In particular, we choose d from $\{30, 50, 100, 200, 350, 500\}$ and conduct the experiments on datasets Office31 and Office-Caltech with the features extracted from the VGG-VD-16 model. The classification results are reported in Table 5. From Table 5, we observe that the results obtained from different dimensions are close to each other for Office-Caltech, and are different from each other for Office 31. Besides, when d is larger than 100, our method tends to perform better. Therefore, as a general guideline for choosing the hyperparameter d, we would suggest picking it between 100 and 350.

4. Conclusion

Under the assumption that the distribution gap results from the between-domain inequality of the joint distributions, the EMMD metric is proposed to measure the discrepancy between the joint distributions. Following that, the JDME method is proposed to learn a mapping to minimize the between-domain EMMD in the projected space such that the performance of classical statistical learning methods can be maintained. Extensive experiments verify the advantages of JDME on both text and image datasets. In the future, we are interested in joint distribution matching with multiple but heterogeneous domains, the distributions of which can be adapted simultaneously via mining their commonness and characteristics. Besides, we are also very interested in extending the current JDME to an end-to-end deep model.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Xiaona Jin: Software, Validation, Formal analysis, Investigation, Visualization, Writing - original draft. **Xiaowei Yang:** Supervision, Writing - review & editing, Funding acquisition. **Bo Fu:** Software, Investigation. **Sentao Chen:** Conceptualization, Methodology, Software, Writing - review & editing.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61906069), Guangdong Basic and Applied Basic Research Foundation (2019A1515011411, 2019A1515011700), Project Funded by China Postdoctoral Science Foundation (2019M662912), Fundamental Research Funds for the Central Universities (2019MS088).

Appendix A. Proof of Proposition 1

Here we prove that if the label kernel $\tau(y_i, y_j)$ satisfies the condition in Proposition 1, then $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)\tau(y_i, y_j)$ is a kernel function for any feature kernel $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. According to Mercer's theorem [41], we only have to prove that any Gram matrix of $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)]$ in the sample space is a semi-positive definite matrix.

The Gram matrix of $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)]$ is

$$\Sigma = \begin{bmatrix} k_{1,1}\tau_{1,1} & \cdots & k_{1,n}\tau_{1,n} \\ \vdots & \ddots & \vdots \\ k_{n,1}\tau_{n,1} & \cdots & k_{n,n}\tau_{n,n} \end{bmatrix}, \tag{31}$$

where $n = n_s + n_t$, $\tau_{i,j} = \sum_{r=1}^{R} u_{ri}u_{rj}$ and u_{ri}, u_{rj} respectively mean the i-th and j-th elements in $\boldsymbol{u}_r$. Σ can be rewritten as

$$\Sigma = \sum_{r=1}^{R} \begin{bmatrix} k_{1,1}u_{r,1}u_{r,1} & \cdots & k_{1,n}u_{r,1}u_{r,n} \\ \vdots & \ddots & \vdots \\ k_{n,1}u_{r,n}u_{r,1} & \cdots & k_{n,n}u_{r,n}u_{r,n} \end{bmatrix} = \sum_{r=1}^{R} \boldsymbol{U}_r^{\top}\boldsymbol{K}\boldsymbol{U}_r, \tag{32}$$

where $\boldsymbol{U}_r = \text{diag}(u_{r1}, \cdots, u_{rn})$, which means the diagonal elements of $\boldsymbol{U}_r$ are made up of the vector $\boldsymbol{u}_r$ and other elements in $\boldsymbol{U}_r$ are all zeros.

Since $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kernel function, for any Gram matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ of $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and any vector $\boldsymbol{z} \in \mathbb{R}^{n_s+n_t}$, we have

$$\boldsymbol{z}\boldsymbol{K}\boldsymbol{z}^{\top} \geqslant 0.$$

Then for any vector $\boldsymbol{v} \in \mathbb{R}^n$, we have

$$\boldsymbol{v}^\top \boldsymbol{\Sigma} \boldsymbol{v} = \sum_{r=1}^{R} (\boldsymbol{U}_r \boldsymbol{v})^\top \boldsymbol{K} (\boldsymbol{U}_r \boldsymbol{v}) \geqslant 0. \quad (34)$$

Because of the arbitrariness of $\boldsymbol{v}$, we can conclude that any Gram matrix $\boldsymbol{\Sigma}$ of $\sigma(\cdot,\cdot)$ is semi-positive definite, and therefore $\sigma[(\boldsymbol{x}_i, y_i), (\boldsymbol{x}_j, y_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)\tau(y_i, y_j)$ is a kernel function.

Appendix B. Proof of the Label Kernel

In this part, we prove that the function $\tau_2(\cdot,\cdot)$ defined in Eq. (13) is a kernel function. According to Mercer's theorem [41], we only have to prove that for any label set $\{y_i^s\}_{i=1}^{n_s} \bigcup \{\hat{y}_i^t\}_{i=1}^{n_t}$, the Gram matrix $\boldsymbol{G} \in \mathbb{R}^{(n_s+n_t)\times(n_s+n_t)}$ of $\tau_2(\cdot,\cdot)$ is a semi-positive definite matrix. In fact, the Gram matrix $\boldsymbol{G}$ can be decomposed into

$$\boldsymbol{G} = \sum_{c=1}^{C} \boldsymbol{G}_c, \quad (35)$$

where $\boldsymbol{G}_c = \boldsymbol{L}_c \boldsymbol{L}_c^\top (\forall c \in \{1, 2, \ldots, C\})$, and $\boldsymbol{L}_c \in \mathbb{R}^{(n_s+n_t)\times 1}$ is defined as

$$\boldsymbol{L}_c(i) = \begin{cases} \frac{n_s}{n_s^c} & \text{if} i \in \mathcal{S} \quad \text{and} \quad y_i = c \\ \frac{n_t}{n_t^c} & \text{if } i \in \mathcal{T} \quad \text{and} \quad y_i = c \\ 0 & \text{if} i \notin \mathcal{C} \end{cases}. \quad (36)$$

It is obvious that all these $\boldsymbol{G}_c(c = 1, \ldots, C)$are semi-positive definite matrices, and therefore their sum $\boldsymbol{G}$ is also a semi-positive definite matrix, which concludes the proof.

Appendix C. Proof of $\tau(\cdot,\cdot)$ Satisfying the Condition of Proposition 1

Since $\tau(y_i, y_j)$ is the sum of $\tau_1(y_i, y_j)$ and $\tau_2(y_i, y_j)$, its Gram matrix in the sample space is also the sum of the Gram matrices of $\tau_1(y_i, y_j)$ and $\tau_2(y_i, y_j)$. According to the proof in Appendix B, the Gram matrix of $\tau_2(y_i, y_j)$ can be decomposed into Eq. (35). For the δ-kernel $\tau_1(y_i, y_j)$, its Gram matrix is

$$\boldsymbol{F} = \sum_{c=1}^{C} \boldsymbol{F}_c, \quad (37)$$

where $\boldsymbol{F}_c = \boldsymbol{M}_c \boldsymbol{M}_c^\top (\forall c \in \{1, 2, \cdots, C\})$, and $\boldsymbol{M}_c \in \mathbb{R}^{(n_s+n_t)\times 1}$ is defined as

$$\boldsymbol{M}_c(i) = \begin{cases} 1 & \text{if} y_i = c \\ 0 & \text{if} y_i \neq c \end{cases}. \quad (38)$$

Then, the Gram matrix of $\tau(y_i, y_j)$ can be calculated as

$$\boldsymbol{E} = \sum_{c=1}^{C} \boldsymbol{L}_c \boldsymbol{L}_c^\top + \sum_{c=1}^{C} \boldsymbol{M}_c \boldsymbol{M}_c^\top, \quad (39)$$

which means that $\tau(y_i, y_j)$ satisfies the condition of Proposition 1.

References

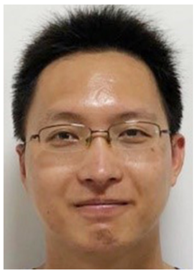
[1] P.A. Absil, R. Mahony, R. Sepulchre, Optimization Algorithms on Matrix Manifolds, Princeton University Press, 2009.
[2] T.A. Almeida, J.M.G. Hidalgo, A. Yamakami, Contributions to the study of sms spam filtering: new collection and results, in: Proceedings of the 11th ACM Symposium on Document Engineering, ACM, 2011, pp. 259–262.
[3] M. Baktashmotlagh, M. Harandi, M. Salzmann, Distribution-matching embedding for visual domain adaptation, J. Mach. Learn. Res. 17 (2016) 3760–3789.
[4] Y. Cao, M. Long, J. Wang, Unsupervised domain adaptation with distribution matching machines, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018..
[5] C.C. Chang, C.J. Lin, Libsvm: a library for support vector machines, ACM Trans. Intell. Syst. Technol. (TIST) 2 (2011) 27.
[6] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: delving deep into convolutional nets. arXiv preprint arXiv:1405.3531, 2014..
[7] S. Chen, L. Han, X. Liu, Z. He, X. Yang, Subspace distribution adaptation frameworks for domain adaptation. IEEE Trans. Neural Networks Learn. Syst. 2020..
[8] S. Chen, X. Yang, Tailoring density ratio weight for covariate shift adaptation, Neurocomputing 333 (2019) 135–144.
[9] X. Chen, S. Wang, M. Long, J. Wang, Transferability vs. discriminability: batch spectral penalization for adversarial domain adaptation, in: International Conference on Machine Learning, 2019a, pp. 1081–1090..
[10] Y. Chen, S. Song, S. Li, C. Wu, A graph embedding framework for maximum mean discrepancy-based domain adaptation algorithms, IEEE Trans. Image Process. 29 (2019) 199–213.
[11] N. Courty, R. Flamary, D. Tuia, A. Rakotomamonjy, Optimal transport for domain adaptation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2016) 1853–1865.
[12] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[13] A. Edelman, T.A. Arias, S.T. Smith, The geometry of algorithms with orthogonality constraints, SIAM journal on Matrix Analysis and Applications 20 (1998) 303–353.
[14] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Unsupervised visual domain adaptation using subspace alignment, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2960–2967.
[15] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. 17 (2016) 2096, 2030.
[16] M.G. Genton, Classes of kernels for machine learning: a statistics perspective, J. Mach. Learn. Res. 2 (2001) 299–312.
[17] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, 2012 IEEE Conference on Computer Vision and Pattern Recognition IEEE (2012) 2066–2073.
[18] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, B. Schölkopf, Domain adaptation with conditional transferable components, International Conference on Machine Learning (2016) 2839–2848.
[19] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, J. Mach. Learn. Res. 13 (2012) 723–773.
[20] H. Hachiya, M. Sugiyama, N. Ueda, Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition, Neurocomputing 80 (2012) 93–101.
[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[22] S. Herath, M. Harandi, F. Porikli, Learning an invariant hilbert space for domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3845–3854.
[23] C.A. Hou, Y.H.H. Tsai, Y.R. Yeh, Y.C.F. Wang, Unsupervised domain adaptation with label and structural consistency, IEEE Trans. Image Process. 25 (2016) 5552–5562.
[24] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, A.J. Smola, Correcting sample selection bias by unlabeled data, Adv. Neural Inf. Process. Syst. (2007) 601–608.
[25] B. Klimt, Y. Yang, Introducing the enron corpus, in: CEAS, 2004..
[26] W.M. Kouw, L.J. Van Der Maaten, J.H. Krijthe, M. Loog, Feature-level domain adaptation, J. Mach. Learn. Res. 17 (2016) 5943–5974.
[27] A. Kumagai, T. Iwata, Unsupervised domain adaptation by matching distributions based on the maximum mean discrepancy via unilateral transformations, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 4106–4113.
[28] J. Li, M. Jing, K. Lu, L. Zhu, H.T. Shen, Locality preserving joint transfer for domain adaptation, IEEE Trans. Image Process. 28 (2019) 6103–6115.
[29] S. Li, C.H. Liu, B. Xie, L. Su, Z. Ding, G. Huang, Joint adversarial domain adaptation, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 729–737.
[30] S. Li, S. Song, G. Huang, Prediction reweighting for domain adaptation, IEEE Trans. Neural Networks Learn. Syst. 28 (2016) 1682–1695.
[31] S. Li, S. Song, G. Huang, Z. Ding, C. Wu, Domain invariant and class discriminative feature learning for visual domain adaptation, IEEE Trans. Image Process. 27 (2018) 4260–4273.
[32] J. Liang, R. He, Z. Sun, T. Tan, Aggregating randomized clustering-promoting invariant projections for domain adaptation, IEEE Trans. Pattern Anal. Mach. Intell. 41 (2018) 1027–1042.
[33] H. Liu, M. Shao, Z. Ding, Y. Fu, Structure-preserved unsupervised domain adaptation, IEEE Trans. Knowl. Data Eng. 31 (2018) 799–812.
[34] Y. Liu, W. Tu, B. Du, L. Zhang, D. Tao, Homologous component analysis for domain adaptation, IEEE Trans. Image Process. 29 (2019) 1074–1089.
[35] M. Long, Y. Cao, J. Wang, M.I. Jordan, Learning transferable features with deep adaptation networks. arXiv preprint arXiv:1502.02791, 2015..
[36] M. Long, Z. Cao, J. Wang, M.I. Jordan, Conditional adversarial domain adaptation, Adv. Neural Inf. Process. Syst. (2018) 1640–1650.
[37] M. Long, J. Wang, G. Ding, J. Sun, P.S. Yu, Transfer feature learning with joint distribution adaptation, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2200–2207.
[38] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017, JMLR. org. pp. 2208–2217..

[39] H. Lu, C. Shen, Z. Cao, Y. Xiao, A. van den Hengel, An embarrassingly simple approach to visual domain adaptation, IEEE Trans. Image Process. 27 (2018) 3403–3417.
[40] L.v.d. Maaten, G. Hinton, Visualizing data using t-sne. J. Mach. Learn. Res. 9 (2008) 2579–2605..
[41] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, philosophical transsaction of the royal society of london, 1909, ser..
[42] S.A. Nene, S.K. Nayar, H. Murase, et al., Columbia object image library (coil-20), 1996..
[43] S.J. Pan, X. Ni, J.T. Sun, Q. Yang, Z. Chen, Cross-domain sentiment classification via spectral feature alignment, in: Proceedings of the 19th international conference on World wide web, ACM, 2010, pp. 751–760..
[44] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (2009) 1345–1359.
[45] S. Roy, A. Siarohin, E. Sangineto, S.R. Bulo, N. Sebe, E. Ricci, Unsupervised domain adaptation using feature-whitening and consensus loss, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9471–9480.
[46] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: European Conference on Computer Vision, 2010, Springer, pp. 213–226.
[47] T. Sim, S. Baker, M. Bsat, The cmu pose, illumination, and expression (pie) database, in: Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition, 2002, IEEE, pp. 53–58..
[48] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014..
[49] B. Sun, J. Feng, K. Saenko, Return of frustratingly easy domain adaptation, in: Thirtieth AAAI Conference on Artificial Intelligence, 2016..
[50] J. Townsend, N. Koep, S. Weichwald, Pymanopt: a python toolbox for optimization on manifolds using automatic differentiation, J. Mach. Learn. Res. 17 (2016) 4755–4759.
[51] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.
[52] J. Wang, Y. Chen, W. Feng, H. Yu, M. Huang, Q. Yang, Transfer learning with dynamic distribution adaptation, ACM Trans. Intell. Syst. Technol. (TIST) 11 (2020) 1–25.
[53] J. Wang, Y. Chen, S. Hao, W. Feng, Z. Shen, Balanced distribution adaptation for transfer learning, in: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 1129–1134.
[54] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, P.S. Yu, Visual domain adaptation with manifold embedded distribution alignment, in: Proceedings of the 26th ACM International Conference on Multimedia, 2018, pp. 402–410.
[55] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, W. Zuo, Mind the class weight bias: weighted maximum mean discrepancy for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2272–2281.
[56] X. Yang, L. Tan, L. He, A robust least squares support vector machine for regression and classification with noise, Neurocomputing 140 (2014) 41–52.
[57] J. Zhang, W. Li, P. Ogunbona, Joint geometrical and statistical alignment for visual domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1859–1867.
[58] Y. Zhang, T. Liu, M. Long, M. Jordan, Bridging theory and algorithm for domain adaptation, International Conference on Machine Learning (2019) 7404–7413.
[59] Z. Zhang, M. Wang, Y. Huang, A. Nehorai, Aligning infinite-dimensional covariance matrices in reproducing kernel hilbert spaces for domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3437–3445.

Xiaona Jin received the B. S. degree in information and computing science from South China University of Technology, Guangzhou, China. She is currently pursuing the M. Sc. degree in software engineering in the School of Software Engineering, South China University of Technology. Her research interests include domain adaptation and computer vision.

Xiaowei Yang received the B. S. degree in theoretical and applied mechanics, the M. Sc. degree in computational mechanics, and the Ph. D. degree in solid mechanics from Jilin University, Changchun, China, in 1991, 1996, and 2000, respectively. He is currently a full time professor in the School of Software Engineering, South China University of Technology. His current research interests include designs and analyses of algorithms for large-scale pattern recognition, imbalanced learning, semi-supervised learning, support vector machines, tensor learning, and evolutionary computation.

Bo Fu received the B. S. degree in software engineering from South China University of Technology, Guangzhou, China. He is currently pursuing the M. Sc. degree in software engineering in the School of Software Engineering, South China University of Technology. His research interests include deep learning and domain adaptation.



Sentao Chen received the B. S. degree in statistics from Guangdong University of Technology, Guangzhou, China. He is currently pursuing the Ph.D. degree in software engineering in the School of Software Engineering, South China University of Technology. His current research interests include statistical machine learning and domain adaptation.