This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

A Unified Framework for Metric Transfer Learning

Yonghui Xu, Sinno Jialin Pan, Hui Xiong, Qingyao Wu, Ronghua Luo, Huaqing Min, and Hengjie Song

Abstract—Transfer learning has been proven to be effective for the problems where training data from a source domain and test data from a target domain are drawn from different distributions. To reduce the distribution divergence between the source domain and the target domain, many previous studies have been focused on designing and optimizing objective functions with the Euclidean distance to measure dissimilarity between instances. However, in some real-world applications, the Euclidean distance may be inappropriate to capture the intrinsic similarity or dissimilarity between instances. To deal with this issue, in this paper, we propose a metric transfer learning framework (MTLF) to encode metric learning in transfer learning. In MTLF, instance weights are learned and exploited to bridge the distributions of different domains, while Mahalanobis distance is learned simultaneously to maximize the intra-class distances and minimize the inter-class distances for the target domain. Unlike previous work where instance weights and Mahalanobis distance are trained in a pipelined framework that potentially leads to error propagation across different components, MTLF attempts to learn instance weights and a Mahalanobis distance in a parallel framework to make knowledge transfer across domains more effective. Furthermore, we develop general solutions to both classification and regression problems on top of MTLF, respectively. We conduct extensive experiments on several real-world datasets on object recognition, handwriting recognition and WiFi location to verify the effectiveness of MTLF compared with a number of state-of-the-art methods.

Index Terms—Transfer learning, Metric learning, Density ratio reweighting, Mahalanobis distance, Learning framework.

1 Introduction

A common assumption underlying many machine learning and data mining algorithms is that training data and test data are represented in the same feature space and drawn from the same distribution. However, this assumption may not hold in many real-world applications, especially when training data come from one domain (a.k.a. the source domain) while the test data are from another domain (a.k.a., the target domain). To address the problem caused by the distribution divergence between the source domain and the target domain, transfer learning has been well studied over the past few years [1], [2].

In practice, transfer learning is desirable to many real-world applications. For example, for sentiment analysis on online product reviews, a main task is to build a classifier to predict the overall sentiment polarity, e.g., positive, neural, or negative, of a given online product review. However, sentiment dictionaries used on different types of products can be very different, resulting in that a sentiment classifier trained on one type of products, i.e., the source domain, may fail to preform well on another type of products, i.e., the target domain [3], [4]. In this case, it would be helpful if classification knowledge can be transferred from the source domain to the target domain. As a second motivating application, for indoor WiFi localization [5], [6], a main task is to predict a mobile user's location based on the received WiFi signals on the mobile device. However, the received WiFi signal strength can vary significantly across different mobile devices even though being received at the same location and at the same time. As a result, a localization model trained on one type of mobile devices may fail to make precise localization on another type of mobile devices. To address this problem, transferring knowledge across different types of mobile devices is a promising solution [7]. As a third example, we consider the problem of object recognition [8], [9]. A main task is to automatically recognize an object from a target scene based on a set of annotated objects from other scenes, a.k.a., source scenes. For this problem, the varying factors (e.g., location and pose, view angle, resolution, motion blur, scene illumination and background clutter between scenes) possibly lead to difference in data distributions between the target scene and the source scenes. From this perspective, how to adapt the recognition models learned from the source scenes to recognize objects from the target scene accurately through knowledge transfer is critical.

Previous studies about transfer learning focused on either re-weighting the instances in the source domain [10], [11], [12], [13] to match the distribution of the target domain, or finding a new representation for the instances in the source domain and the target domain [4], [14], [15], [16], [17] in order to reduce the distribution divergence between the source domain and the target domain. As far as we know, most existing transfer learning methods utilize the Euclidean distance to measure the dissimilarity between instances in the source domain or the target domain. This may make knowledge transfer suffer from the limitations associated with the Euclidean metric. Specifically, from the perspective of classification, the objective functions that are described using the Euclidean distance may be subopti-

- *Yonghui Xu, Qingyao Wu, Ronghua Luo, Huaqing Min, and Hengjie Song are with South China University of Technology, China E-mail: hqmin@scut.edu.cn, sehjsong@scut.edu.cn.*
- *Sinno Jialin Pan is with School of Computer Science and Engineering, Nanyang Technological University, Singapore.*
- *Hui Xiong is with Management Science and Information Systems Department, Rutgers, the State University of New Jersey, USA.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

mal: commonly they would not maximize the inter-class distances while minimizing intra-class distances [18], [19], [20], [21]. An analogous problem also exists in regression problems because the Euclidean distance may not be able to capitalize on any statistical regularities in the data that might be estimated from a large training set of labeled data [19]. In addition, if most of the original input features are irrelevant to the target regression or classification task, then using a pre-defined and data-independent distance is not able to learn a precise model. As a result, how to reduce distribution divergence between domains and learn an appropriate distance for the target domain simultaneously is crucial for transfer learning.

To address these issues, we propose a metric transfer learning framework (MTLF) to encode metric learning into transfer learning. On one hand, MTLF employs instance weights to bridge the difference in distributions between the source domain and the target domain. On the other hand, a Mahalanobis distance is utilized in MTLF to maximize the distance between classes and minimize the distance within each class. Our main contributions are two folds.

1) Different from many instance-based transfer learning approaches that only focus on learning instance weighs for the source domain data with the Euclidean distance, MTLF takes advantage of a learned Mahalanobis distance (i.e., unit less, scale-invariant and taking into account the correlations of the datasets), and thus can more effectively preserve and utilize the intrinsic geometric information among the instances of different domains with similar or dissimilar labels. In this way, MTLF improves the accuracy of the target domain tasks using the reweighted instances in the source domain.
2) Different from some metric-based transfer learning approaches that learn the instance weights for the source domain data and a Mahalanobis distance metric for the target domain data in a pipelined framework, in MTLF, an alternating optimization method is proposed to learn the instance weights and a Mahalanobis distance metric simultaneously. In this way, the knowledge, i.e., the weights for source domain data, which is used as a bridge across domains, can be learned more precisely under a more appropriate distance metric.

The rest of this paper is organized as follows. In the following section, we briefly review some work related to our study. In Section 3, we present the proposed framework, MTLF, and two general solutions to regression and classification problems on top of MTLF in detail, respectively. In Section 4, we propose an alternating optimization method to solve MTLF in particular. We report the experimental results on several benchmark datasets in Section 5. Finally, we conclude this paper and discuss future work in Section 6.

2 Related Work

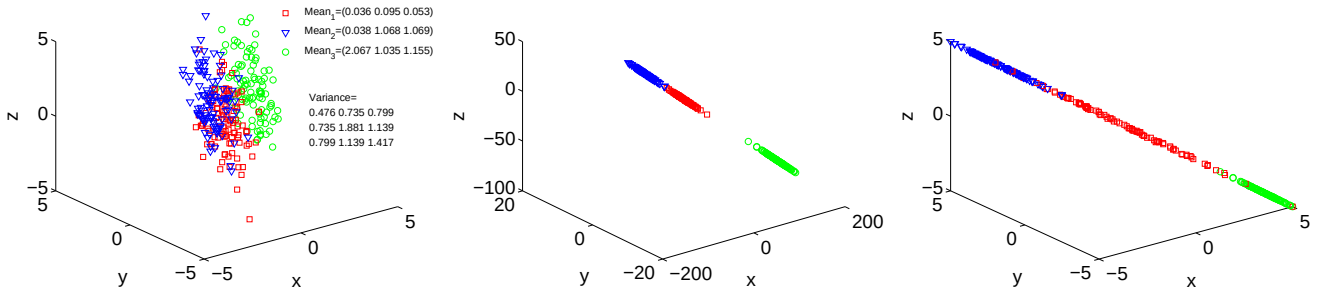
Recently, various transfer learning techniques have been proposed for transferring knowledge in terms of instances, features, parameters or relational information among data objects across domains [1], [2]. Among existing transfer learning methods, approaches related to our study can be summarized into three categories: *instance-based transfer learning*, *feature-based transfer learning with distribution matching* and *metric-based transfer learning*.

As there are a number of transfer learning approaches in the first category [13], here, we just briefly review some representatives. TrAdaBoost [11] adjusts the weights of the labeled data in the source domain to filter out the instances, which are most unlikely drown from the target domain distribution. By doing so, the re-weighted source-domain instances compose a distribution similar to the one found at the target domain. Finally, the re-weighted instances are treated as additional training data to learn a model for the target domain. Since the traditional TrAdaBoost is only applicable to classification problems, Pardoe and Stone [22] extended TrAdaBoost and proposed ExpBoost.R2 and TrAdaBoost.R2 to deal with regression problems. Wan *et al.* [23] proposed the Bi-weighting domain adaptation (BIW) algorithm for cross-language text classification. BIW first aligns the feature spaces of the source domain and the target domain into the same coordinate system, and then adjusts the instance and feature weights of the training data in the source domain.

In the second category, the goal is to match the distributions between the source domain and the target domain based on subspace learning or feature learning. For instance, Pan *et al.* [16], [24] proposed learn a latent space for the source domain and the target domain, where the distance in distributions between domains can be minimized while some important properties of the data can be preserved. Their proposed methods utilize the techniques on kernel embedding of distributions to learn the latent space, and solve the proposed optimization problem using either semidefinite programming or generalized eigen decomposition. A follow-up work proposed by Si *et al.* [15] adopts a different criterion to measure distance between distributions and solve the resultant optimization problem using gradient-descent-based approaches. Shao *et al.* [25] attempted to impose a low-rank constraint to match the source domain and the target domain in the subspace for transfer learning. Baktashmotlagh *et al.* [26] proposed a subspace-based method and a sample selection method for unsupervised domain adaptation by exploiting the probability distributions that lie on a Riemannian manifold. Ding *et al.* [27] developed a Deep Low-Rank Coding (DLRC) by integrating feature learning and knowledge transfer in a unified deep learning framework. By exploiting a discriminative low-rank coding, DLRC attempts to mitigate the marginal and conditional distributions between source domain and the target domain. Long *et al.* [28] proposed to reduce the difference between domains in both the marginal distribution and the conditional distribution through a dimensionality reduction procedure. Though a common subspace or a new feature representation can be obtained by these methods, the potential impact of Mahalanobis distance in preserving the intrinsic geometric information of the instances in different domains is ignored.

In the third category, Zhang and Yeung [29] proposed the transfer metric learning (TML) algorithm to learn a metric for the target task by exploiting the correlations between the target task and a set of source tasks. Kulis *et*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering



(a) Original data (b) b (c) c

Fig. 1. An example to demonstrate different impacts on classification with the Euclidean distance and a Mahalanobis distance respectively. (a) shows the original data generated according to three Gaussian distributions with different means. (b) shows the data obtained by transforming the three Gaussian distributions data with a learned Mahalanobis distance. (c) indicates the transformed data with unified scale as in (a).

al. [30] proposed a kernel-based method, ARC-t, to learn a transformation between domains based on metric learning. Fouad *et al.* [31] proposed an approach to learning with the so-called *privileged information* through metric learning in prototype-based models. However, these metric-based learning methods do not explicitly reduce difference, e.g., the divergence in distributions, between domains in the learning process. Recently, a metric learning method based on deep learning was proposed [32], namely Deep Transfer Metric Learning (DTML). In DTML, though a Maximum Mean Discrepancy (MMD) [33] based regularization term is introduced to reduce the distance in distributions between domains, a deep neural network is used to approximate the feature map used in MMD instead of a characteristic kernel [31]. In this way, minimization on the distance estimated by MMD may not be able to significantly reduce the inter-domain distance in theory.

In theory, to represent an arbitrary distribution using kernel embedding, a kernel needs to be characteristic or universal [33]. MMD is a non-parametric measure to estimate distance between distributions using kernel embedding. If the kernel is not characteristic, there is no guarantee that the estimated distance between distributions is precise though empirically, non-characteristic kernels might be used to obtain good performance for transfer learning [16]. In DTML, the neural network excluding its top layer can be considered as an implicit nonlinear feature map ϕ to map raw data to a "high-level" feature space, and in the top layer, a linear kernel $k(\phi(x_i), \phi(x_j)) = \langle \phi(x_i), \phi(x_j) \rangle$ w.r.t. $\phi(x)$ is used for MMD. If there is a proof that the kernel $k(\phi(x_i), \phi(x_j))$ is a characteristic kernel w.r.t x, then the MMD theory can be ensured, and thus the distance between the source domain distribution and the target domain distribution can be estimated precisely, otherwise, minimization on the estimated distance might not be able to significantly reduce the inter-domain distance in the theory.

The most related work to our study is the consistent distance metric learning method (CDML) proposed by Cao *et al.* [34]. By investigating two weight strategies (i.e., instance weights and instance-pair weights), CDML attempts to solve the metric learning problem under covariate shift [35]. Note that, although the basic ideas behind CDML and our proposed MTLF are similar, i.e., to learn a distance metric for the target domain by reducing the distribution divergence between the source domain and the target domain, MTLF differs from CDML in two aspects: 1) MTLF is a general framework to solve both classification and regression problems, while CDML is only applicable to classification problems, 2) In CDML, instance weights and a Mahalanobis distance metric are learned in a pipelined framework where the instance weights are first learned under the Euclidean metric, and a Mahlanobis distance metric is then trained with the learned instance weights. In contrast, in MTLF, we propose an alternating optimization approach to learn the instance weights and a distance metric simultaneously. By doing so, MTLF can match distributions between the source domain and the target domain more effectively (or precisely) and hence learn a more suitable Mahalanobis distance for the target domain task.

3 Metric Transfer Learning Framework

3.1 Problem Statement

Denoted by $D_S = \{(x_j, y_j)|j = 1, ..., N_S\}$ the source-domain labeled data, where $x_j \in R^{d \times 1}$ is an input feature vector, and $y_j \in R$ is the corresponding output, and $D_T = D_T^l \bigcup D_T^u = \{(x_j, y_j)|j = N_S + 1, ..., N_S + N_T^l\} \bigcup \{x_j|j = N_S + N_T^l + 1, ..., N_S + N_T\}$ the partially labeled data in the target domain, where $N_T^l \ll N_S$. In our transfer learning setting, the marginal distributions of the source and target domain data are different, i.e., $P_T(x) \neq P_S(x)$. As a result, a distance metric learned with D_S may not be desired for D_T, while the labeled data in D_T is too few to learn a precise distance metric. Therefore, our goal is to learn a Mahalanobis distance metric for the target domain with D_T by adaptively exploiting the labeled data from D_S.

3.2 The Overall Framework

Learning a Mahalanobis distance metric has been shown to be useful for many classification and regression problems [36], [37]. Fig. 1 shows an example to demonstrate different impacts on classification with the Euclidean distance metric and a Mahalanobis distance metric respectively. Fig. 1(a) shows the original data generated according to three Gaussian distributions with different means. From this figure, we can observe that it is not easy to distinguish the data objects drawn from different distributions under the Euclidean distance. Fig. 1(b) shows the data obtained by transforming the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

three Gaussian distributions data with a Mahalanobis distance (i.e., the Mahalanobis distance learned by information-theoretic metric learning [38]). In Fig. 1(b), it is easy to distinguish the data objects drawn from different distributions under the Mahalanobis distance. Fig. 1(c) shows the transformed data with unified scale as in Fig. 1(a). We can observe that different distributions can easily be distinguished under the learned Mahalanobis distance, even with unified scale.

In this paper, we focus on learning a Mahalanobis distance metric for the target domain. Let $M \in R^{d\times d}$ be a positive semi-definite matrix, a Mahalanobis distance metric between a pair of instances x_i and x_j is defined as follows,

$$d_{ij} = \sqrt{(x_i - x_j)^\top M (x_i - x_j)}. \tag{1}$$

As M is positive semi-definite, it can be decomposed as $M = A^\top A$, where $A \in R^{d\times d}$. Therefore, learning a Mahalanobis distance in terms of M is equivalent to learn a matrix A.

Because the distributions of the source domain and the target domain are different, $P_S(x) \neq P_T(x)$, the source-domain labeled data cannot be used directly to learn a distance metric for the target domain. Similar to most instance-based transfer learning approaches, here we assume that some labeled data in the source domain after reweighting are still able to provide discriminative information for the target domain, and thus useful for learning a metric for the target domain. Therefore, in this work, we offer a unified framework to learn instance weights ω for the source domain data, a Mahalanobis distance metric A for the target domain, and a final predictive model f for the target domain, simultaneously. The proposed unified framework is written as follows,

$$\min_{A,\omega,f} \mathcal{J} = r(A) + \lambda\psi(\omega) + \beta\ell(f, A, \omega; D_S, D_T^l), \tag{2}$$

where $\lambda > 0$ and $\beta > 0$ are tradeoff parameters to balance impact of different terms in the objective. The regularization term $r(A)$ controls the generalization error of the metric in terms of A. Here, we define $r(A)$ as the following form,

$$r(A) = \text{tr}(A^\top A). \tag{3}$$

The second term in the objective (2), i.e., $\psi(\omega)$, is the regularization term on the instance weights for the source-domain labeled data. To estimate the instance weights, all the source and target domain data including labeled and unlabeled are used. The introduction of this regularization term is to avoid some potential issues in learning the instance weights ω and the Mahalanobis distance metric A, which will be discussed in Section 3.3. The third term in the objective (2) is the loss function of the predictive model f with the learned metric A on the target-domain labeled data as well as the re-weighted source-domain labeled data. Specific formulations of the loss function for classification and regression problems will be defined in the Section 3.4 and Section 3.5, respectively.

3.3 A Specific Form of $\psi(\omega)$

In this paper, we define the form of the regularization term $\psi(\omega)$ as follows,

$$\psi(\omega) = \|\omega - \omega_0\|^2, \tag{4}$$

where $\omega_0(x_i) = \frac{P_T(x_i)}{P_S(x_i)}$ is the estimated density ratio or weight of an instance x_i in the source domain under the Euclidean metric. The higher is the value of $\omega_0(x_i)$, the higher is the value of $P_T(x_i)$ and/or the smaller is the value of $P_S(x_i)$. This implies that x_i is more similar to the target domain distribution than the the source domain distribution, and thus is of more importance to the target domain, under the Euclidean metric.

To estimate the density ratio $\frac{P_T(x)}{P_S(x)}$, we adopt the approach proposed by [39], where the density ratio is considered as a function that can be approximated by a linear combination of some basic functions, i.e., $\omega_0(x_i) = \sum_{j=1}^{b} \alpha_j \phi_j(x_i)$, where $\{\phi_j\}$'s indicate a set of pre-defined basis functions and $\{\alpha_j\}$'s represent the corresponding nonnegative parameters to be learned. Different setting of ϕ_j may affect the performance of the density ratio estimation. Herein, we follow [39] to define the basic function ϕ_j using a Gaussian kernel function centered at c_j,

$$\phi_j(x) = \exp\left\{-\frac{\|x - c_j\|^2}{\sigma^2}\right\}, \tag{5}$$

where c_j can be chosen from a subset of instances in the target domain. The weights of the source domain instances, $\omega_0(x)$, can be obtained by minimizing the KL-divergence between $P_T(x)$ and $\omega_0(x)P_S(x)$ as follows,

$$\begin{aligned}\min_{\omega_0} &\text{KL}\left(P_T(x)\|\omega_0(x)P_S(x)\right) \\ &= \int P_T(x)\log\frac{P_T(x)}{\omega_0(x)P_S(x)}\,dx \\ &= \int P_T(x)\log\frac{P_T(x)}{P_S(x)}\,dx - \int P_T(x)\log\omega_0(x)\,dx. \end{aligned}\tag{6}$$

As shown in [39], (6) can be converted to the following optimization problem,

$$\begin{aligned}\max_{\alpha} \quad & \sum_{x_j\in D_T} \log\sum_{i=1}^{b}\alpha_i\phi_i(x_j) \\ s.t. \quad & \sum_{x_j\in D_S}\sum_{i=1}^{b}\alpha_i\phi_i(x_j) = N_S, \text{ and } \alpha > 0.\end{aligned}\tag{7}$$

Since (7) is convex, gradient ascent approaches can be applied to obtain the optimal solution.

Note that in (7), we utilize the source-domain labeled data and all target domain data including both labeled and unlabeled to learn the parameters, $\{\alpha_j\}$'s. Then the prior of the weight of each source-domain labeled data can be calculated by $\omega_0(x_i) = \sum_{j=1}^{b}\alpha_j\phi_j(x_i)$, which is further used to learn a more precise weight $\omega(x_i)$ with the regularization term (4). In this way, distribution divergence between the reweighted source domain data and the target domain data can be minimized.

For simplicity in presentation, in the rest of this paper, we denote by $\hat{\omega}_0 \in R^{(N_S+N_T^l)\times 1}$ the instance weight vector with $\hat{\omega}_0(x_i) = \omega_0(x_i)$ for $x_i \in D_S$ and $\hat{\omega}_0(x_i) = 1$ for $x_i \in D_T^l$. Accordingly, we denote by $\hat{\omega} \in R^{(N_S+N_T^l)\times 1}$ the instance weight vector with $\hat{\omega}(x_i) = \omega(x_i)$ for $i = 1, ..., N_S$ to be learned, while $\hat{\omega}(x_i) = 1$ for $i > N_S$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

3.3.1 Discussion on the Regularization Term

In some previous metric-based transfer learning methods, e.g., CDML [34], instance weights and a Mahalanobis distance metric are learned in a pipelined framework. To reduce the difference between D_S and D_T, the vector of instance weights ω_0 is first obtained under the Euclidean distance by minimizing the KL-divergence as discussed in (6). Then in the second step, a Mahalanobis distance metric in terms of A is learned for D_T by transferring supervised information from D_S through the learned weight vector ω_0.

A major drawback in such a pipelined framework is that the vector of instance weights ω_0 is learned under the Euclidean distance without taking an appropriate distance metric into consideration. In this case, the bridge between domains constructed through ω_0 may not be precise. As a result, the error introduced in learning ω_0 can propagate to the learning on A. This limits the power of knowledge transferred across domains.

Suppose that A' and A are the ideal distance metric for the source domain and the target domain respectively, then the ideal instance weights ω to be learned should be

$$\omega(x) = P_T^A(x)/P_S^{A'}(x), \tag{8}$$

where $P_S^{A'}(x)$ and $P_T^A(x)$ are density estimations of D_S and D_T under the distance metric A' and A, respectively, which is different from $\omega_0(x) = \frac{P_T(x)}{P_S(x)}$. However, optimizing A and ω by directly minimizing the KL-divergence, $\mathrm{KL}(P_T^A(x)||\omega(x)P_S^{A'}(x))$, is intractable[1]. Therefore, in this paper, we propose to use ω_0 as a prior for learning ω rather than enforcing $\omega = \omega_0$ as proposed in CDML.

3.4 MTLF for Regression Problems

When addressing regression problems, we first define the following form for the predictive model f,

$$f(x_i) = \frac{1}{Z} \sum_{j \neq i}^{N_S+N_T^l} k_{ij}^A y_j, \tag{9}$$

where Z is a normalization term,

$$Z = \sum_{j \neq i}^{N_S+N_T^l} k_{ij}^A,$$

and k^A is referred to as the kernel function under the distance metric A to calculate similarity between instances. In this paper, we use a Gaussian kernel for k^A, i.e., $k_{ij}^A = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right)$, where d_{ij} is the Mahalanobis distance in terms of A between instances i and j, and σ is the parameter of the Gaussian kernel.

Note that (9) is adapted from kernel regression [36] by calculating the kernel matrix using a Mahalanobis distance metric. We further define the loss function $\ell(f, A, \hat{\omega}; D_S, D_T^l)$ as the sum of weighted squared errors as follows,

$$\ell(f, A, \hat{\omega}, D_S, D_T) = \sum_{i=1}^{N_S+N_T^l} \hat{\omega}(x_i) \|f(x_i) - y_i\|^2. \tag{10}$$

1. Note that A' can be learned in advance for the source domain data.

By substituting (3), (9) and (10) into the unified framework (2), we can obtain the specific optimization problem for regression as follows,

$$\begin{aligned} \min_{A,\hat{\omega}} \quad & \mathrm{tr}(A^\top A) + \lambda \|\hat{\omega} - \hat{\omega}_0\|^2 \\ & + \beta \sum_{i=1}^{N_S+N_T^l} \hat{\omega}(x_i) \left\| \frac{1}{Z} \sum_{j \neq i}^{N_S+N_T^l} k_{ij}^A y_j - y_i \right\|^2 \\ s.t. \quad & \sum_{i=1}^{N_S} \hat{\omega}(x_i) = N_S, \text{ and } \hat{\omega}(x_i) \geq 0, \end{aligned} \tag{11}$$

where the constraints are used to normalize $\hat{\omega}$.

3.5 MTLF for Classification Problems

When tackling classification problems, we adapt the approach proposed by [36] to use K-nearest-neighbor with the instance weights $\hat{\omega}$ and under the distance metric A as a classifier by defining the loss function as follows,

$$\ell(f, A, \hat{\omega}, D_S, D_T) = \ell_{in}(A, \hat{\omega}) - \ell_{out}(A, \hat{\omega}), \tag{12}$$

where

$$\begin{cases} \ell_{in}(A, \hat{\omega}) = \sum_{y_i = y_j} \hat{\omega}(x_i)\hat{\omega}(x_j)\|A(x_i - x_j)\|^2 \\ \\ \ell_{out}(A, \hat{\omega}) = \sum_{y_i \neq y_j} \hat{\omega}(x_i)\hat{\omega}(x_j)\|A(x_i - x_j)\|^2, \end{cases}$$

where $\ell_{in}(A, \hat{\omega})$ is the sum of the weighted difference within class and $\ell_{out}(A, \hat{\omega})$ is the sum of the weighted difference between classes, under the distance metric A, respectively. By substituting (12) and (3) into (2), we obtain the specific optimization problem for classification as follows,

$$\begin{aligned} \min_{A,\hat{\omega}} \quad & \mathrm{tr}(A^\top A) + \lambda \|\hat{\omega} - \hat{\omega}_0\|^2 \\ & + \beta \sum_{i,j} \hat{\omega}(x_i)\hat{\omega}(x_j)\|A(x_i - x_j)\|^2 \delta_{ij} \\ s.t. \quad & \sum_{i=1}^{N_S} \hat{\omega}(x_i) = N_S, \text{ and } \hat{\omega}(x_i) \geq 0, \end{aligned} \tag{13}$$

where δ_{ij} is an indicator function, where

$$\delta_{ij} = \begin{cases} 1, \ y_i = y_j, \\ -1, \ y_i \neq y_j. \end{cases}$$

Note that in (13) we use all the in-class and out-of-class instance-pairs to estimate the loss function value. However, a larger number of such instance-pairs, denoted by C, may increase the computational cost. The computational complexity of our algorithm is $O(Cd^2)$. To balance the accuracy and the time cost, in practice, cross-validation can be used to choose a tradeoff value for C.

4 Optimization

In this section, we derive approaches to solve the optimization problems constructed in (11) and (13), respectively. For the sake of simplicity, we first derive an overall optimization approach for the following unified optimization problem, which is constructed by using $\ell(f, A, \hat{\omega}, D_S, D_T)$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

as a general loss function for (11) and (13), and converting the optimization problem to an unconstrained one,

$$\min_{A,\hat{\omega}} \mathcal{J} = r(A) + \lambda\|\hat{\omega}-\hat{\omega}_0\|^2 + \beta\ell(f,A,\hat{\omega},D_S,D_T) \quad (14)$$
$$+\rho\left((\hat{\omega}^T e - N_S)^2 + \sum_{i=1}^{N_S}(\max(0,-\hat{\omega}(x_i)))^2\right),$$

where ρ is a nonnegative penalty coefficient and $e \in R^{(N_S+N_T^l)\times 1}$, where $e_i = 1$ if $i \le N_S$, and $e_i = 0$ if $N_S < i \le N_S + N_T^l$.

As both the classification and regression models are constructed in a non-parametric form, we do not need to optimize (14) with respect to f explicitly. Therefore, we propose an alternating optimization algorithm to learn A and $\hat{\omega}$ alternatingly and iteratively. To be specific, at the t-th iteration, we first fix the matrix A_t and update the value of $\hat{\omega}_t$ using gradient descent based on the following rule,

$$\hat{\omega}_{t+1} = \hat{\omega}_t - \gamma_1 \left.\frac{\partial\mathcal{J}}{\partial\hat{\omega}}\right|_{\hat{\omega}_t}, \quad (15)$$

where $\gamma_1 > 0$ is an adaptive step-size. The derivative of the objective $\mathcal{J}$ w.r.t. $\hat{\omega}$ in (15) can be written as

$$\frac{\partial\mathcal{J}}{\partial\hat{\omega}} = 2\lambda(\hat{\omega}-\hat{\omega}_0) + \beta\zeta + \rho[2(\hat{\omega}^T e - N_S)e + \hat{\omega}^2\xi],$$

where ξ is the vector with $\xi_i = \text{sign}(\max(0,-\hat{\omega}(x_i)))$, and ζ is the vector with

$$\zeta_i = \begin{cases} \left\| \frac{1}{Z}\sum_{j\neq i}^{N_S+N_T^l} k_{ij}^A y_j - y_i \right\|^2, & \text{for regression,} \\ \sum_{i,j}\hat{\omega}(x_j)\|A(x_i-x_j)\|^2\delta_{ij}, & \text{for classification.} \end{cases}$$

After updating the value of $\hat{\omega}_{t+1}$, we then alternatingly fix $\hat{\omega}_{t+1}$ and update A_t based on the following rule,

$$A_{t+1} = A_t - \gamma_2 \left.\frac{\partial\mathcal{J}}{\partial A}\right|_{A_t}, \quad (16)$$

where $\gamma_2 > 0$ is an adaptive step-size. For regression problems, the derivative of the objective $\mathcal{J}$ w.r.t. A can be written as

$$\begin{aligned}
&\frac{\partial\mathcal{J}}{\partial A} \\
&= 2A + 2\beta\sum_{i=1}^{N}\hat{\omega}(x_i)(\hat{y}_i - y_i)\frac{\partial\hat{y}_i}{\partial A} \\
&= 2A + \\
&\quad 2\beta\sum_{i=1}^{N}\frac{\hat{\omega}(x_i)(\hat{y}_i-y_i)}{Z}\left(\sum_{j\neq i}^{N} y_j\frac{\partial k_{ij}^A}{\partial A} - \hat{y}_i\sum_{j\neq i}^{N}\frac{\partial k_{ij}^A}{\partial A}\right) \\
&= 2A + 2\beta\sum_{i=1}^{N}\frac{\hat{\omega}(x_i)(\hat{y}_i-y_i)}{Z}\sum_{j\neq i}^{N}(\hat{y}_i - y_j)\frac{\partial k_{ij}^A}{\partial A} \\
&= 2A + 4\beta A\sum_{i=1}^{N}\frac{\hat{\omega}(x_i)(\hat{y}_i-y_i)}{Z}\sum_{j\neq i}^{N}(y_j-\hat{y}_i)k_{ij}^A v_{ij}v_{ij}^T,
\end{aligned}$$

Algorithm 1 The MTLF algorithm

Require: Source-domain labeled data D_S, target-domain labeled data D_T^l, target-domain unlabeled data D_T^u, initializations A_0 and $\hat{\omega}^0$, step sizes γ_1 and γ_2, tradeoff parameters λ and β, penalty conefficient σ, threshold ε, and maximum number of iterations T.

1: **procedure** MTLF(A, $\hat{\omega}$)
2: **for** $t := 0$ to T **do**
3: Compute the gradient $\frac{\partial\mathcal{J}(A,\hat{\omega}^t)}{\partial A}$, and $\frac{\partial\mathcal{J}(A_t,\hat{\omega})}{\partial\hat{\omega}}$.
4: Update $\hat{\omega}$ by $\hat{\omega}^{t+1} = \hat{\omega}^t - \gamma_1 \left.\frac{\partial\mathcal{J}(A_t,\hat{\omega})}{\partial\hat{\omega}}\right|_{\hat{\omega}^t}$.
5: Update A by $A_{t+1} = A_t - \gamma_2 \left.\frac{\partial\mathcal{J}(A,\hat{\omega}^{t+1})}{\partial A}\right|_{A_t}$.
6: **if** $\left|\mathcal{J}(A_{t+1},\hat{\omega}^{t+1}) - \mathcal{J}(A_t,\hat{\omega}^t)\right| < \varepsilon$ **then**
7: $A = A_{t+1}$, $\hat{\omega} = \hat{\omega}^{t+1}$, break.
8: **end if**
9: **end for**
10: **end procedure**

Ensure: $(A,\hat{\omega}) = \arg\min\mathcal{J}(A,\hat{\omega})$

where $N = N_S + N_T^l$, $\hat{y}_i = f(x_i)$, and $v_{ij} = x_i - x_j$. For classification problems, the derivative of the objective $\mathcal{J}$ w.r.t. A can be written as,

$$\frac{\partial\mathcal{J}}{\partial A} = 2\beta\sum_{i,j}\hat{\omega}(x_i)\hat{\omega}(x_j)Av_{ij}v_{ij}^T\delta_{ij} + 2A.$$

We alteratingly and iteratively update $\hat{\omega}$ and A until the change in values of the objective function $\mathcal{J}$ is less than a predefined threshold ε. The algorithm is summarized in Algorithm 1. Note that the initialization of matrix A_0 can be learned in advance from the source domain, and the instance weight vector $\hat{\omega}_0$ can be initialized based on the Euclidean distance.

4.1 Computational Complexity

In this section, we analyze the computational complexity of our proposed MTLF. We denote by T the number of iterations. The computational cost for computing the gradient of $\mathcal{J}$ with respect A is $O(T(N_S+N_T^l)Cd^2)$, where C is the number of instance-pairs, and d is the dimensionality of each instance. The computational cost for computing the gradient of $\mathcal{J}$ with respect $\hat{\omega}$ is $O(TCd^2)$. Moreover, the computational cost caused by other operations, i.e., Lines 4-7 in Algorithm 1 is $O(TCd^2)$. Therefore, the overall computational complexity of Algorithm 1 is $O(T(N_S+N_T^l)Cd^2)$.

5 Experiments

To verify the effectiveness of the proposed MTLF [2], we conduct extensive experiments on both regression and classification problems on serval benchmark datasets, including indoor localization on the WiFi Dataset [5] [3], object recognition on the Office-Caltech Dataset [30] [4], and handwriting recognition on the USPS [5]-MNIST [6] Dataset. The details of these datasets are described in the following section.

2. Source codes are available at https://github.com/xyh2016/MTLF
3. http://www.cse.ust.hk/~qyang/ICDMDMC07/
4. http://www.eecs.berkeley.edu/~jhoffman/domainadapt/
5. http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html
6. http://yann.lecun.com/exdb/mnist/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Statistics of the three datasets.

Datasets	Tasks	Dimensions	Categories	Labeled instances in SD (D_S)	Labeled instances in the TD (D_T^l)	Test instances in TD (D_T^u)
WiFi Dataset	WiFi	101	-	621	53	3,075
Office-Caltech Dataset	a-c	800	10	200	30	1,093
	w-a	800	10	80	30	928
	w-d	800	10	80	30	127
	w-c	800	10	80	30	1,093
	d-w	800	10	80	30	265
	d-c	800	10	80	30	1,093
	c-a	800	10	80	30	928
	c-w	800	10	80	30	265
	c-d	800	10	80	30	127
USPS-MNIST Dataset	usps-mnist	256	10	9,298	1,000	10,000
	mnist-usps	256	10	10,000	1,000	8,298

5.1 Datasets

Table 1 shows some statistics of the three datasets, i.e., the WiFi dataset, the Office-Caltech dataset and the USPS-MNIST dataset, used in our experiments.

5.1.1 WiFi Dataset

The WiFi dataset is commonly used to evaluate the performance of transfer learning methods for regression [16], which is about an indoor WiFi localization problem over time. All the training and test data are collected around a floor in a building. It consists of 621 training instances in the source domain (i.e., D_S) and 3,128 labeled instances (including 53 instances for training (i.e., D_T^l) and 3,075 instances for testing (i.e., D_T^u) in the target domain, where each instance is represented by 101 features.

5.1.2 Office-Caltech Dataset

The Office-Caltech dataset is the benchmark dataset for cross-domain object recognition which contains 10 overlapping categories from 4 domains: Amazon (a), Webcam (w), Dslr (d) and Caltech256 (c). Images in the domains, Amazon and Caltech256, are from amazon.com and office environment, respectively. The domains, Webcam and Dslr, contain images taken by a webcam and a dslr camera, respectively. The images in different domains are taken with varying factors (*i.e.*, location and pose, view angle, resolution, motion blur, scene illumination and background clutter between scenes) which leads to difference in distributions between the four domains. From this dataset, we construct 9 cross-domain recognition tasks for experiments, each of which is denoted by *SD-TD*, where *SD* denotes the source domain and *TD* denotes the target domain. For example, the task a-c denotes that Amazon is used as the source domain and Caltech256 is used as the target domain. For the tasks with Amazon as the source domain, 20 instances per category in Amazon are randomly selected as D_S, and 3 labeled instances per category in the target domain are randomly selected as D_T^l. For all the other tasks, 8 instances per category in the source domain are randomly selected as D_S and 3 labeled instances per category in the target domain are randomly selected as D_T^l.

5.1.3 USPS-MNIST Dataset

The USPS dataset and the MNIST dataset are widely used in computer vision and pattern recognition. The USPS dataset consists of 9,298 labeled images, each of which is of the size of 16×16. The MNIST dataset consists of 60,000 training images and 10,000 test images, each of which is of the size of 28×28. Note that, The USPS and MNIST datasets are subject to different distributions and both contain 10 categories. We construct two handwriting recognition tasks *usps-mnist* and *mnist-usps* based on these two datasets. For example, the task *usps-mnist* denotes that USPS is used as the source domain and MNIST is used as the target domain. For the task *usps-mnist*, 9,298 labeled instances in the source domain are selected as D_S, 1,000 labeled instances in the target domain are randomly selected as D_T^l, and 10,000 test images in the target domain are used for test, i.e., D_T^u. For the task *mnist-usps*, 10,000 labeled instances in the source domain are selected as D_S, 1,000 labeled instances in the target domain are randomly selected as D_T^l, and 8,298 test images in the target domain are used for test, i.e., D_T^u. By considering the fact that MTLF is designed for the case where the source domain and the target domain have the same feature space, we rescale all images to the size of 16×16.

5.2 Comparison Baselines

We compare our proposed MTLF with a number of state-of-the-art methods. On the WiFi dataset, the baseline methods for comparison include a support vector regression machine (SVR) and MLKR [36], which is a regression model with a data-dependent distance metric, AdaBoost.R2 and TrAdaBoost.R2 [22] which are transfer learning methods for regression. On the Office-Caltech dataset, the baseline methods for comparison include support vector machines, SVMs with source-domain labeled only and SVMt with target-domain labeled data only, transfer learning methods based on metric learning, ARC-t and CDML, and feature-based transfer learning methods, HFA [40], MMDT [41], GFK [42], JDA [28] and SISS [26]. On the USPS and MNIST datasets, the baseline methods for comparison include support vector machines (SVM with both source-domain and target-domain labeled data, SVMs, and SVMt), CDML and MMDT. Note that all these baseline models except for SVMs and SVMt are trained on the labeled data in both the source and target domains, while they are tested on the unlabeled data in target domains.

Regarding our proposed MTLF, to further investigate the impact of the instance weights and the learned Mahalanobis distance to the overall performance, we denote by MTLF_1 a reduction of MTLF only using instance weights and fixing A to be an identity matrix, by MTLF_2 a reduction of MTLF only using the learned Mahalanobis distance and fixing all the instance weights to be one, and by MTLF_3 the proposed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

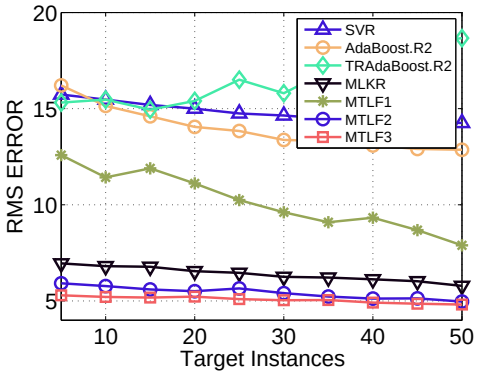


Fig. 2. Experimental results on the WiFi dataset.

MTLF using both the instance weights and the learned Mahalanobis distance.

To make fair comparisons, all algorithms adopt the same experimental setup. In particular, all the algorithms were implemented with MATLAB R2013a and run on a machine with Windows OS. All the experiments were conducted over 20 random permutations for each dataset, and the results are reported by averaging over the 20 runs.

5.3 Regression on WiFi Dataset

In this experiment, we first analyze the impact of different components of MTLF, and then compare MTLF with other state-of-the-art methods. In the first experiment, we compare the effectiveness of the learned instance weights and Mahalanobis distance for MTLF. Experimental results are shown in Fig. 2. From the figure, we can find that in the application of indoor WiFi-based localization, learning an appropriate Mahalanobis metric contributes more significantly to the overall prediction results in terms of root mean squared error (RMSE) than learning instance weights for the source-domain labeled data. Moreover, learning instance weights with an appropriate metric can further improve prediction results. We also note that more target-domain labeled data used in training can lead to smaller RMSE. Compared to MTLF_2 and MTLF_3, the performance of MTLF_1 in terms of RMSE is much more sensitive to the size of the target-domain labeled data.

In Fig. 2, we further compare the prediction results of MTLF_3 with those of other baseline methods including MLKR, SVR, AdaBoost.R2 and TrAdaBoot.R2. From the figure, we find that MLKR and MTLF_3 perform much better than SVR, AdaBoost.R2 and TrAdaBoost.R2 in terms of RMSE. This is mainly because that both MLKR and MTLF_3 learn a Mahalanobis distance metric for the WiFi data instead of using the Euclidean distance. As we analyzed in the first experiment, an appropriate Mahalanobis distance metric is more effective to model WiFi data. With an inappropriate metric, the transferred labeled data from the source domain may even hurt the performance of the target model as shown by TrAdaBoost.R2's performance under varying size of target-domain labeled data. The reason why MTLF_3 outperforms MLKR is that with the learned metric, MTLF_3 further learn instance weights to reduce the difference in distributions between domains. Note that the results on the WiFi dataset shown in Figure 2 are different from those reported in [16]. This is because as shown in [16], a de-noising step is important for WiFi data. However, in this paper we focus on evaluating the effectiveness of a metric learned by MTLF_3, thus we do not perform any de-noising process on the WiFi data.

To further demonstrate the effectiveness of MTLF, in Fig. 3 we visualize the location map covered by MTLF and MLKR from the test data in the target domain, respectively. As can be seen from the figure, the map recovered by MTLF is much closer to the ground-truth map than that covered by MLKR. As we discussed, this is mainly because MLKR does not take the difference between domains into consideration when learning the distance metric. Note that the map recovered by MTLF in 3(c) is not very smooth. This is because as shown in [43], the intrinsic structure behind WiFi data may be a manifold, while MTLF does not exploit the manifold structure to learn the distance metric and instance-weights. In our future work, we will study how to encode a manifold structure of data into MTLF.

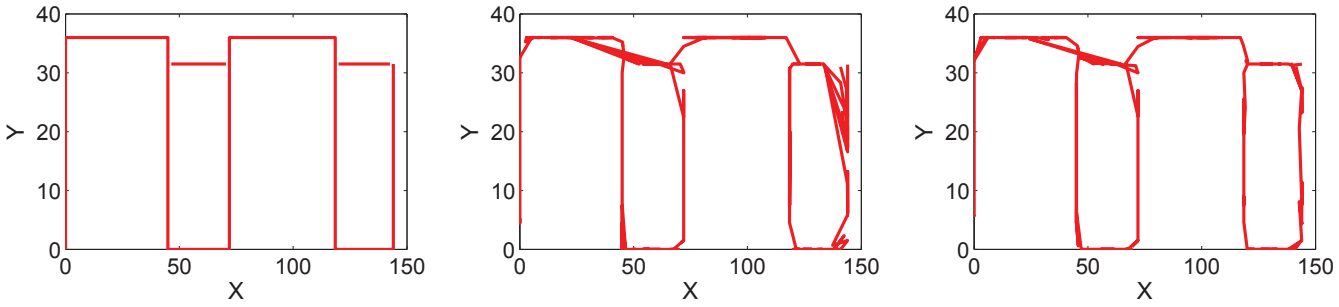
5.4 Classification on Office-Caltech Dataset

In our experiments, we learn a Mahalanobis distance metric via MTLF, and use a KNN Classifier for classification, where the distance between two instances is defined as (1). For a test image x_i, we first compute the learned Mahalanobis distance between x_i and each source-domain and target-domain labeled instance. Then we chose the K nearest instances to x_i as the references. The class labels of x_i is determined by a majority vote of the K nearest reference instances.

Experimental results on the Office-Caltech dataset are presented in Table 2. To make a fair comparison, we use $\sigma/\sqrt{n_{trail}}$ instead of σ in Table 2, where σ is the standard deviation, and $n_{trail} = 20$ is the number of random permutations. From Table 2, we can observe that similar to the results shown in Figure 2, the performance of MTLF_2 and MTLF_3 in terms of classification accuracy is much better than MTLF_1. Furthermore, MTLF_3 consistently outperforms MTLF_2. This is because in MTLF_1, the matrix A is replaced by an identity matrix, which results in that the predictions made by MTLF_1 is equivalent to those made by a K-nearest-neighbor classifier under the Euclidean distance. In this case, instance weights are ineffective. While in MTLF_2, the weights of all instances including both the source- and target- domain instances are equal, which fails to reduce the difference between domains. From the results, we can conclude that to transfer supervised information across domains for object recognition effectively, learning weights for source-domain labeled data and an appropriate metric for the target domain data need to be done simultaneously.

From Table 2, we can also find that compared to non-transfer classifiers i.e., SVMs and SVMt, transfer learning methods i.e., GFK, ARC-t, MMDT, CDML and MTLF_3, can obtain much better performance in terms of recognition accuracy. Among all the transfer learning methods, MTLF_3 performs best on the 9 tasks on average. This is because MMDT, HFA and ARC-t do not take the potential importance of reducing the divergence in distributions between the source domain and the target domain into consideration, and thus do not minimize the divergence in distributions

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering



(a) GroundTruth. (b) MLKR. (c) MTLF.

Fig. 3. Recovered location maps from the WiFi dataset.

TABLE 2
Experimental results on the Office-Caltech dataset (Accurancy±Standard deviation/$\sqrt{n_{trail}}$ in %).

Groups	GFK	HFA	SVMt	SVMs	ARC-t	MMDT	CDML	JDA	SISS	MTLF1	MTLF2	MTLF3
a-c	36.0±0.5	31.1±0.6	32.0±0.8	35.1±0.3	37.0±0.4	36.4±0.8	38.1±0.5	37.5±0.3	38.3±0.4	15.9±0.5	36.5±0.3	**38.5±0.3**
w-a	44.1±0.4	45.9±0.7	45.6±0.7	35.7±0.4	43.4±0.5	**47.7±0.9**	41.9±0.4	43.5±0.4	44.0±0.5	20.7±0.5	44.1±0.5	**45.5±0.4**
w-d	70.5±0.7	51.7±1.0	55.1±0.8	66.6±0.7	71.3±0.8	67.0±1.1	65.7±0.6	69.4±0.8	71.8±0.9	17.8±1.1	69.7±0.8	**73.3±0.7**
w-c	31.1±0.6	29.4±0.6	30.4±0.7	31.3±0.4	31.9±0.5	32.2±0.8	33.5±0.3	33.5±0.3	33.2±0.5	12.4±0.5	33.6±0.5	**34.6±0.5**
d-w	76.5±0.5	62.1±0.7	62.1±0.8	74.3±0.5	78.3±0.5	74.1±0.8	69.5±0.7	78.6±0.5	**82.8±0.6**	27.6±1.1	80.1±0.4	81.7±0.5
d-c	32.9±0.5	31.0±0.5	31.7±0.6	31.4±0.3	33.5±0.4	34.1±0.8	**35.2±0.2**	34.4±0.4	34.8±0.4	15.1±0.5	34.0±0.4	34.6±0.5
c-a	44.7±0.6	45.5±0.9	45.3±0.9	35.9±0.4	44.1±0.6	44.6±0.8	42.0±0.6	43.1±0.4	46.3±0.6	27.2±0.5	44.5±0.4	**48.1±0.6**
c-w	63.7±0.8	60.5±0.9	60.3±1.0	30.8±1.1	55.9±1.0	63.8±1.1	44.5±0.9	52.8±1.1	59.3±1.6	29.6±0.9	61.0±1.1	**63.7±1.4**
c-d	57.7±1.1	51.9±1.1	55.8±0.9	35.6±0.7	50.6±0.8	56.5±0.9	42.5±0.9	46.8±1.0	51.1±1.2	31.3±1.1	52.6±0.6	**58.8±0.3**
Mean	50.8	45.5	46.5	41.9	49.6	50.7	45.9	48.8	51.3	22.0	50.7	**53.2**

explicitly. As a result, they may fail to fully and effectively transfer knowledge from the source domain to the target domain. In contrast, $MTLF_3$ explicitly reduces divergence in distributions between domains by employing a regularization term to learn weights for the source-domain labeled data. Unlike the other baselines, SISS attempts to minimize the distance between the re-weighted source distribution and the target distribution, JDA tries to learn a new feature representation to jointly match both the marginal distributions and conditional distributions. However, as SISS and JDA both ignore the advantage of distance metric learning, and the distributions matching may be conducted under an inappropriate distance metric.[7] Furthermore, compared to CDML, $MTLF_3$ achieves better performance. This is because CDML suffers from the limitation caused by the pipelined framework. In contrast, $MTLF_3$ learns instance weights and a Mahalanobis distance metric simultaneously, and thus overcome the limitation caused by the pipelined framework. By this way, the metric learned by $MTLF_3$ is supposed to be more appropriate and useful for the cross-domain classification tasks.

5.5 Classification on USPS-MNIST Dataset

On the USPS-MNIST dataset, we compare our proposed MTLF with MMDT, CDML, SVM, SVMs, SVMt in terms of accuracy on large scale handwriting recognition. The experimental results are reported in Table 3.

From the table, we observe that the performances of CDML of SVM are even worse than that of SVMt, which only uses target-domain labeled data for training. This is because if the distributions of the source domain and the target domain are very different under the Euclidean distance, then using the source-domain label in a brute-force manner as SVMt does may get very poor performance. Furthermore, in this case, the error in matching distributions under the Euclidean distance may be large, which will be further propagated to the metric learning step, resulting in poor performance. We also observe that MTLF performs better than the other transfer learning baseline MMDT. This is because on the USPS-MNIST dataset, there is little background and texture information that can be exploited by MMDT to construct a powerful representation for the source domain data and the target domain data. In contrast, MTLF performs distribution matching through instance reweighting under a data-dependent metric instead of finding a new feature representation. The learned Mahalanobis metric is still able to capture and utilize intrinsic geometric structure among the instances effectively, even with little background and texture information.

TABLE 3
Comparison results on the USPS-MNIST dataset (Accuracy in %).

	SVMs	SVMt	SVM	MMDT	CDML	MTLF
usps-mnist	18.12	65.88	62.22	82.03	56.53	**82.99**
mnist-usps	13.08	81.9	68.17	85.86	57.21	**90.1**

More detailed comparison results in terms of accuracy on different classes on the USPS-MNIST dataset are reported in Table 4. From the table, we can find that none of the four comparison methods achieves the best results on all the classes. Moreover, the baseline methods perform very unstably on different classes. These observations suggest that without distributions matching between domains under a proper distance metric, the classification performance may

7. Note that the results of SISS and JDA shown in Table 2 are different from those reported in the original papers [26] and [28]. This may be because that the labeled instances of the source and the target domains are randomly selected for training (20 random runs), which may result in difference of experimental settings between ours and theirs.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
Experimental results of case study on the USPS-MNIST dataset in accuracy%.

Tasks	Methods	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
usps-mnist	SVM	**95.61**	67.22	42.73	56.83	47.56	63.34	73.7	70.72	73.82	32.01
	CDML	76.94	92.07	73.45	21.78	71.49	28.48	65.24	50.68	23.61	53.91
	MMDT	92.45	95.24	**84.5**	**77.43**	77.09	66.48	81.84	**86.48**	**75.05**	80.08
	MTLF	93.37	**99.21**	72.77	75.84	**79.43**	**71.64**	**90.61**	85.31	72.18	**86.62**
mnist-usps	SVM	88.06	**99.56**	74.18	38.86	44.91	9.27	65.07	79.22	65.57	79.15
	CDML	88.43	95.82	43.42	37.93	21.25	28.55	49.59	69.74	40.49	48.47
	MMDT	96.79	96.69	**88.73**	83.16	72.14	70.7	92.19	91.63	**87.21**	64.67
	MTLF	**98.21**	99.39	85.95	**88.06**	**76.21**	**79.67**	**95.07**	**92.33**	83.44	**89.77**

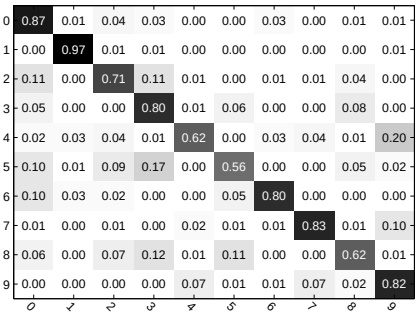


Fig. 4. Confusion matrix of MTLF on the mnist-usps task.

be sensitive on different classes or datasets. In contrast, as shown in the confusion matrix of MTLF on the mnist-usps task in Fig. 4, MTLF performs stably on differently classes, which verifies the robustness of MTLF.

5.6 Parameter Tuning

In this section, we conduct sensitivity studies on the parameters of MTLF. In the general optimization objective of MTLF for both classification and regression (14), there are two trade-off parameters λ and β, and one penalty coefficient. Furthermore, regarding the specific solution to classification, we have two additional parameters, the number of nearest neighbors used in KNN, K, and the number of instance pairs, C. Besides these parameters involved in MTLF, for preprocessing on high-dimensional data, e.g., images of the Office-Caltech dataset and the USPS-MNIST dataset, we use principal component analysis (PCA) [44]. Therefore, we also conduct experiments on the sensitivity of the reduced dimensionality d. In the following experiments, when we study the sensitivity on some parameter(s), we fiex the values for the other parameters.

5.6.1 Sensitivity Study on K for KNN Classifiers

Fig. 5(a) reports the potential impacts imposed by different numbers of nearest neighbors, K, to the accuracy of MTLF on the Office-Caltech dataset. From the figure, we can observe that the accuracies of MTLF on the tasks d-w and w-d are much higher than those of the other tasks with varying values of K. We also find that the accuracies of MTLF on the tasks d-w, w-d and w-a decrease with the increasing values of K, while the accuracies of MTLF on the tasks a-c, w-c and d-c change slightly under varying values of K. These results show that a large value of K may reduce the accuracy of cross-domain classification problems, while a small value of K may perform more stably and better. These empirical observations suggest us to set K to be a small integer, e.g., $K = 1$.

5.6.2 Sensitivity Study on Number of Instance Pairs C

As the C instance pairs are randomly chosen for optimization in classification problems, in this experiment, we verify the impact of different values of C to the overall accuracy of MTLF. Experimental results with varying values of C ranging from 50 to 1,500 are shown in Fig. 5(b). From the figure we can observe that the accuracies of MTLF on the 6 tasks change slightly with increasing values of C. Based on these empirical observations, we set the number of instance pairs, C, to 100 for classification. We do not set a larger value for C because a larger value of C causes higher cost in terms of computational time. Note that the influence of different values of C to the cost of MTLF in terms of computational time will be further analyzed in Section 5.6.5.

5.6.3 Sensitivity Study on Penalty Coefficient ρ

In Section 4, we formulate the constrained problems (11) and (13) as an unconstrained problem (14) by adding a penalty term. Accordingly, ρ in (14) is the penalty coefficient of the penalty term. Theoretically, it is not difficult to find that a larger ρ may make us more assurance to obtain the optimal solutions of A and $\hat{\omega}$ [45]. However, a larger ρ may make the computation in practical applications more difficult [45]. To choose a suitable ρ, we conduct two experiments.

We conduct the sensitivity analysis on ρ on on the Office-Caltech dataset, whose experimental results are shown in Fig. 5(c). From the figure, we find that when $\rho \geq 1$, the performance of MTLF is smooth and stably good, and a larger value of ρ when $\rho > 1$ does not lead to a significantly higher accuracy. To analyze the impact of different values of ρ on the convergence rate of our proposed algorithm, we conduct another set of experiments on the d-w task by varying the values of ρ from 1 to 10. Fig. 6 shows the experimental results, where $\Delta\mathcal{J} = \mathcal{J}_{t+1} - \mathcal{J}_t$, $\Delta A = \|A_{t+1} - A_t\|$ and $\Delta\hat{\omega} = \|\hat{\omega}^{t+1} - \hat{\omega}^t\|$. The x-axis in Fig. 6(a), Fig. 6(b) and Fig. 6(c) represents the number of iterations t. From the figures, we observe that for each value of ρ, $\Delta\mathcal{J}$, ΔA and $\Delta\hat{\omega}$ all consistently decrease after several iterations. We also find that at an iteration t, a larger value of ρ always lead to a larger value of $\Delta\mathcal{J}$, ΔA or $\Delta\hat{\omega}$, which implies that more iterations are required for convergence. This suggests that if we choose a very large value for ρ, we may suffer from expensively computational time for convergence. By considering the results shown in Fig. 6(c) that the accuracies of MTLF with $\rho \geq 1$ are very similar, we can set ρ to be a relatively small value, e.g., $\rho = 1$ or $\rho = 2$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering

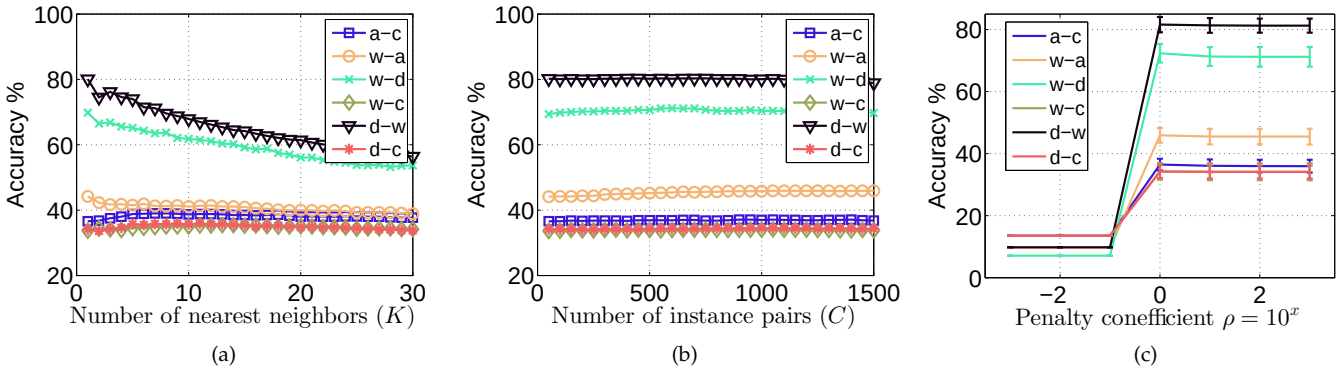


Fig. 5. Sensitivity analysis of the number of nearest neighbors K, number of instance pairs C, and penalty coefficient ρ of MTLF on the Office-Caltech dataset.

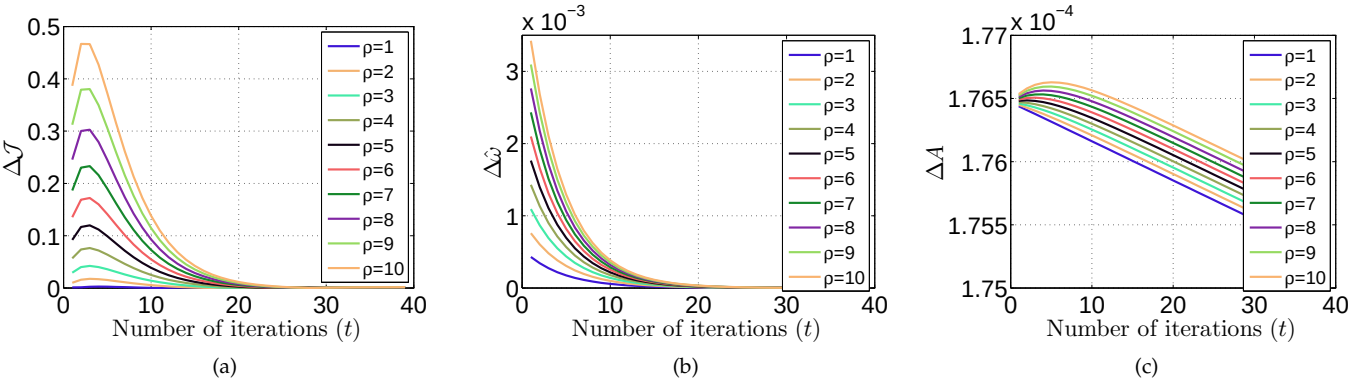


Fig. 6. Impact of different ρ's on the convergence rate for MTLF.

5.6.4 Sensitivity Study on β and λ

Here, we study sensitivity analysis on the two trade-off parameters, β and λ, in the objective of MTLF (14). Experiments are conducted on the *mnist-usps* task of the USPS-MNIST dataset, whose results are shown in Fig. 7 From the figure, we observe that MTLF performs well and stably when β is set to be a relatively small value, i.e., $\beta \leq 0.01$, and λ is in a wide range $[10^{-3}\ 10^{3}]$.

Tradeoff β

Tradeoff λ	0.001	0.01	0.1	1	10	100	1000
0.001	89.36	90.06	80.92	36.84	17.91	6.44	8.64
0.01	89.36	90.06	80.92	36.84	17.91	6.44	8.64
0.1	89.36	90.06	80.92	36.84	17.91	6.44	8.64
1	89.36	90.06	80.92	36.84	17.91	6.44	8.64
10	89.36	90.06	80.92	36.84	17.91	6.44	8.64
100	89.36	90.06	80.92	36.84	17.91	6.44	8.64
1000	89.36	90.06	80.92	36.84	17.91	6.44	8.64

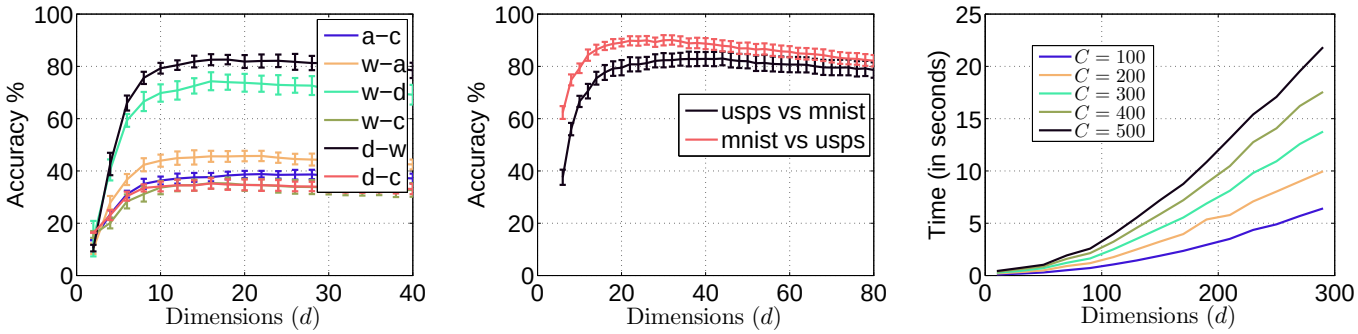
Fig. 7. Sensitivity analysis on β and λ.

5.6.5 Sensitivity Study on Dimensionality d

Note that in our experiments on the Office-Caltech dataset and the USPS-MNIST dataset, we use principal component analysis (PCA) as a preprocessing method to transform the data from a high-dimensional space to a low-dimensional space. In this experiment, we test the influence of different dimensions d to accuracy and computational time of MTLF, respectively. As the dimension of the WiFi data is small, we use all features of the original data for experiments on the WiFi dataset.

Fig. 8(a) reports the experimental results of MTLF on the Office-Caltech dataset with varying values of d. In this experiment, we change d from 2 to 40. From the figure we can observe that MTLF performs well and stably on all the tasks when $d \geq 10$. The experimental results of MTLF on the USPS-MNIST dataset with varying values of d from 3 to 80 are reported in Fig. 8(b). From the figure, we can observe that MTLF performs well and stably on all the tasks when $20 \leq d \leq 30$. Fig. 8(c) reports the computational time on the a-c task of the Office-Caltech dataset with varying numbers of dimension d and varying numbers of instance pairs, C. The x-axis represents the dimension, and the y-axis represents the computational time. Curves of different colors represent the computational time corresponding to different values of C, respectively. This experiment is tested on a computer deployed with a dual-core CPU (2.0 GHz) and 8.0GB RAM memory. From the figure, we can observe that the computational time increases with the increasing values of d, and a larger value of C leads to longer computational time. The experimental results shown in Fig. 8(a), Fig. 8(b) and Fig. 8(c) suggest that we can set $d \in [20\ 30]$ to achieve good performance and computational efficiency.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2017.2669193, IEEE Transactions on Knowledge and Data Engineering



(a) Influence of dimensions d to MTLF on Office-Caltech dataset.

(b) Influence of dimensions d to MTLF on USPS-MNIST dataset.

(c) Influence of number of instance pairs, C and dimensions d to the time cost of MTLF.

Fig. 8. Influence of different d's for MTLF.

5.6.6 Discussion

Based on the above empirical experiments, we summarize the suggested values or ranges for all the parameters of the proposed MTLF in Table 5, which can be considered as priors to use cross-validation to tune optimal parameter settings on specific datasets.

TABLE 5
Suggested values or ranges for parameters.

K	C	ρ	β	λ	d
1	100	1 or 2	0.01	1	[20 30]

6 Conclusion and Future Work

In this paper, we present a metric transfer learning framework (MTLF) to address classification and regression problems in a unified framework. By defining the objective functions on the basis of a Mahalanobis distance, instead of the Euclidean distance, MTLF makes it possible to more efficiently preserve and utilize the intrinsic geometric information among the instances from different domains with similar/dissimilar labels. With these advantages, MTLF can maximize the inter-class distances and minimize the intra-class distances for the target domain. In this way, MTLF improves the accuracy of the target domain task with the reweighted instances in the source domain. In addition, we discuss the limitation of learning a Mahalanobis distance and instance weights in a pipelined framework. To overcome this limitation, we propose an alternating optimization method to learn them simultaneously. As a result, MTLF is able to transfer knowledge from the source domain to the target domain more effectively.

Experiments on real-world regression or classification problems verify the superiority of MTLF over other state-of-the-art methods. Though MTLF has shown promising results for transfer learning problems, there are still some open issues to be studied. For example, how to apply the proposed MTLF to imbalanced data classification problems [46] or multi-instance multi-label classification problems [47]. In these problem settings, it is difficult to match the distributions between domains, which makes it challenging to learn an optimal Mahalanobis distance for the target domain. In our future work, we would like to extend our framework to solve these problems.

Acknowledgment

Professor Huaqing Min and Professor Hengjie Song are the corresponding authors. Sinno J. Pan thanks the support of the NTU Singapore Nanyang Assistant Professorship (NAP) grant M4081532.020. Hengjie Song thanks the support of the National Natural Science Foundation of China (ID:71671069), the Natural Science Foundation of Guangdong Province, China (ID:2016A030313479), the Mobile Research Foundation of Ministry of Education of China (ID:MCM20160204), and the Fundamental Research Funds for the Central Universities. Huaqing Min thanks the support of the Guangzhou Key Laboratory of Robotics and Intelligent Software under Grant No.15180007.

References

[1] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345 – 1359, 2010.

[2] S. J. Pan, "Transfer learning," in *Data Classification: Algorithms and Applications*, 2014, pp. 537–570.

[3] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, 2007.

[4] S. J. Pan, X. Ni, J.-T. Sun, Q. Yang, and 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.

[5] Q. Yang, S. J. Pan, and V. W. Zheng, "Estimating location using wi-fi," *IEEE Intelligent Systems*, no. 1, pp. 8–13, 2008.

[6] J. J. Pan, S. J. Pan, J. Yin, L. M. Ni, and Q. Yang, "Tracking mobile users in wireless networks via semi-supervised colocalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 587 – 600, 2012.

[7] V. W. Zheng, S. J. Pan, Q. Yang, and J. J. Pan, "Transferring multi-device localization models using latent multi-task learning," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 1427–1432.

[8] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proceedings of the 11th European Conference on Computer Vision*, 2010, pp. 213–226.

[9] J. Donahue, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell, "Semi-supervised domain adaptation with instance constraints," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013, pp. 668 – 675.

[10] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, "Correcting sample selection bias by unlabeled data," in *Advances in Neural Information Processing Systems*, 2006, pp. 601–608.

[11] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 193–200.

[12] Y. Tsuboi, H. Kashima, S. Hido, S. Bickel, and M. Sugiyama, "Direct density ratio estimation for large-scale covariate shift adaptation." *Information and Media Technologies*, vol. 4, no. 2, pp. 529–546, 2009.

[13] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[14] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006, pp. 120–128.

[15] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 7, pp. 929–942, 2010.

[16] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199 – 210, 2011.

[17] X. Shi, Q. Liu, W. Fan, and P. S. Yu, "Transfer across completely different feature spaces via spectral embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 906–918, 2013.

[18] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 157–165, 2006.

[19] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, 2005, pp. 1473–1480.

[20] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," *Michigan State Universiy*, vol. 2, 2006.

[21] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu, "Adaptation regularization: A general framework for transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2014.

[22] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 863 – 870.

[23] C. Wan, R. Pan, and J. Li, "Bi-weighting domain adaptation for cross-language text classification," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011, pp. 1535–1540.

[24] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, 2008, pp. 677–682.

[25] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," *International Journal of Computer Vision*, vol. 109, no. 1-2, pp. 74–93, 2014.

[26] M. Baktashmotlagh, M. Harandi, B. Lovell, and M. Salzmann, "Domain adaptation on the statistical manifold," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2481–2488.

[27] Z. Ding, M. Shao, and Y. Fu, "Deep low-rank coding for transfer learning," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 3453–3459.

[28] M. Long, J. Wang, G. Ding, J. Sun, and P. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2200–2207.

[29] Y. Zhang and D. Yeung, "Transfer metric learning by learning task relationships," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 1199–1208.

[30] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 1785–1792.

[31] S. Fouad, P. Tiňo, S. Raychaudhury, and P. Schneider, "Incorporating privileged information through metric learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 7, pp. 1086–1098, 2013.

[32] J. Hu, J. Lu, and Y.-P. Tan, "Deep transfer metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2015.

[33] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[34] B. Cao, X. Ni, J.-T. Sun, G. Wang, and Q. Yang, "Distance metric learning under covariate shift," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011, pp. 1204–1210.

[35] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[36] K. Q. Weinberger and G. Tesauro, "Metric learning for kernel regression," in *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, 2007, pp. 612–619.

[37] B. Kulis, "Metric learning: A survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287 – 364, 2012.

[38] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 209–216.

[39] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation," in *Advances in Neural Information Processing Systems*, 2008, pp. 1433–1440.

[40] L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for heterogeneous domain adaptation," in *Proceedings of the 29th International Conference on Machine Learning*, vol. 1, 2012, pp. 711 – 718.

[41] J. Hoffman, E. Rodner, J. Donahue, T. Darrell, and K. Saenko, "Efficient learning of domain-invariant image representations," *arXiv preprint arXiv:1301.3224*, 2013.

[42] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2066 – 2073.

[43] J. J. Pan, S. J. Pan, J. Yin, L. M. Ni, and Q. Yang, "Tracking mobile users in wireless networks via semi-supervised colocalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 587–600, 2012.

[44] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[45] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.

[46] S. Ando, "Classifying imbalanced data in distance-based feature space," *Knowledge and Information Systems*, pp. 1–24, 2015.

[47] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *Artificial Intelligence*, vol. 176, no. 1, pp. 2291–2320, 2012.

Yonghui Xu received his B.Sc. degree in information and computing science from the Henan University, Kai Feng, China, in 2011. He is currently a Ph.D. student with the School of Software Engineering, South China University of Technology, Guangzhou, China. His current research interests are focused on transfer learning, metric learning, multi-instance multi-label learning and their applications in computer vision and bioinformatics engineering.

Sinno Jialin Pan is a Nanyang Assistant Professor with the School of Computer Science and Engineering at Nanyang Technological University (NTU), Singapore. Prior to joining NTU, he was a scientist and lab head of text analytics with the Data Analytics Department, Institute for Infocomm Research, Singapore. He received his Ph.D. degree in computer science from the Hong Kong University of Science and Technology in 2010. His research interests include transfer learning and its real-world applications.

Hui Xiong (SM07) received the BE degree from the University of Science and Technology of China (USTC), China, the MS degree from the National University of Singapore (NUS), Singapore, and the PhD degree from the Univer-sity of Minnesota (UMN). He is currently a professor and vice chair of the Management Science and Information Systems Department, and the director of Rutgers Center for Information Assurance at the Rutgers, the State University of New Jersey, where he received a two-year early promotion/tenure in 2009, the Rutgers University Board of Trustees Research Fellowship for Scholarly Excellence in 2009, and the ICDM-2011 Best Research Paper Award in 2011. His general area of research is data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications. He has published prolifically in refereed journals and conference proceedings (three books, more than 40 journal papers, and more than 60 conference papers). He is a coeditor-in-chief of Encyclopedia of GIS, an associate editor of IEEE TKDE, and the KAIS journal. He has served regularly on the organization and program committees of numerous conferences, including as a program co-chair of the Industrial and Government Track for KDD-2012 and a program co-chair for ICDM-2013. He is a senior member of the ACM and IEEE.

Qingyao Wu received his B.Sc. degree at the South China University of Technology in 2007, and the M.Sc. degree at Shenzhen Graduate School, Harbin Institute of Technology in 2009, and the Ph.D. degree in the Department of Computer Science, Shenzhen Graduate School, Harbin Institute of Technology. He is currently an Associate Professor with the School ofof Software Engineering, South China University of Technology. His research interests include data mining, machine learning and bioinformatics, particularly in multi-label learning and ensemble learning.

Ronghua Luo received his B.Sc. degree and M.S. degree in material science from the Harbin Institute of Technology, Harbin, China, in 1998 and 2001 respectively, and the Ph.D. degree in Computer Science from Harbin Institute of Technology, in 2006. He is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology. His current research interests include intelligent robotics and robot vision.

Huaqing Min received the Ph.D. degree in computer software and theory from Huazhong University of Science and Technology, Wuhan, China, in 1998. He is currently a Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. He is also the head of the Guangzhou Key Laboratory of Robotics and Intelligent Software. His current research interests are focused but not limited to robotics, intelligent software and automated systems. He also serves as a committee member of RoboCup in the China Chapter, and makes active contributions to the robot soccer community.

Hengjie Song received his B.Sc. degree in E.E in 1998, and the M.Sc. degree in aerospace computer design in 2002 from the Harbin Institute of Technology, and the PhD candidate from tht Nanyang Technological University (NTU), Singapore. He is currently a Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His research interests are focused on data mining and artificial intelligence. He has served as a PC-reviewer for KDD-2015.