A general domain specific feature transfer framework for hybrid domain adaptation

Wei, Pengfei; Ke, Yiping; Goh, Chi Keong

2019

Wei, P., Ke, Y., & Goh, C. K. (2018). A general domain specific feature transfer framework for hybrid domain adaptation. IEEE Transactions on Knowledge and Data Engineering, 31(8), 1440-1451. doi:10.1109/TKDE.2018.2864732

<https://hdl.handle.net/10356/138954>

<https://doi.org/10.1109/TKDE.2018.2864732>

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: <https://doi.org/10.1109/TKDE.2018.2864732>.

Downloaded on 26 Apr 2023 20:43:09 SGT

A General Domain Specific Feature Transfer Framework for Hybrid Domain Adaptation

Pengfei Wei¹, Yiping Ke¹, Chi Keong Goh²
 Nanyang Technological University, Singapore¹,
 Rolls-Royce Advanced Technology Centre, Singapore²
 {pwei001, ypke}@ntu.edu.sg, ChiKeong.Goh@Rolls-Royce.com

Abstract—Heterogeneous domain adaptation needs supplementary information to link up different domains. However, such supplementary information may not always be available in real cases. In this paper, a new problem setting called hybrid domain adaptation is investigated. It is a special case of heterogeneous domain adaptation, in which different domains share some common features, but also have their own domain specific features. We leverage upon common features instead of supplementary information to achieve effective adaptation. We propose a general domain specific feature transfer framework, which can link up different domains using common features and simultaneously reduce domain divergences. Specifically, we learn the translations between common features and domain specific features. Then we cross-use the learned translations to transfer the domain specific features of one domain to another domain. Finally, we compose a homogeneous space in which the domain divergences are minimized. We instantiate the general framework to a linear case and a nonlinear case. Extensive experiments verify the effectiveness of the two cases.

Index Terms—Knowledge transfer, domain specific feature, hybrid domain adaptation.

1 INTRODUCTION

IN conventional machine learning, the test data is assumed to be drawn from the same distribution with the training data so that the model trained from the training set can generalize satisfactory performance on the test set [1]. However, such an assumption does not always hold in reality. In review sentiment analysis, for instance, a sentiment model is trained from reviews of books, but the test is done on reviews of electronic products which have different vocabularies from those of books [2]. The data distribution discrepancy in the training and test sets leads to a dilemma. On the one hand, the labeled data in the test set is too scarce to train a model, and on the other hand, the performance of directly applying the training model to the test set leaves much to be desired.

Domain adaptation (DA) [3], [4] provides an effective way to resolve the dilemma. The objective of DA is to learn a model that works well in a target domain (refers to the test set) where none or scarce labeled data is available, by leveraging upon the knowledge from a different but related source domain (refers to the training set) with plenty of labeled data. It has been widely studied in many applications, for instance, sentiment analysis [2], [5], [6], object recognition [7], [8], [9], and text classification [10], [11], [12].

Depending on the homogeneity of feature spaces, DA can be categorized into the homogeneous one and the heterogeneous one. In homogeneous DA, the feature spaces of two domains are the same, but the feature distributions are different. We still take the review sentiment analysis as an example. Two sets of sentiment reviews are commenting on the same product books, but come from two different merchant webs, e.g. Amazon and Taobao. In this case, although these two sets share the same vocabulary, the words distribution of the two sets may differ considerably due to the customer preference of the two merchants. Compared to homogeneous DA, heterogeneous DA is more difficult to tackle due to the heterogeneity of feature spaces, e.g. different vocab-

ularies for reviews of books and electronic products. Since no correspondence exists between the feature spaces of the two domains, supplementary information, such as labels [13], translators [14], or correspondence instances [15], is needed to link up the two domains. If such information is unavailable, which is usually the case in reality, heterogeneous DA is intractable.

In practice, there are many cases where the feature spaces of two domains are not completely heterogeneous. Two domains share some common features, while each domain also has its own domain specific features. In sentiment analysis, for instance, reviews of different products share some common sentiment words, e.g., “good”, “bad”, etc. In addition, different types of products have their specific sentiment words, e.g., “comprehensive” for books, “noisy” for electronic products, etc. Similarly in biological data, different species may share common traits or characteristics but also have their own biological features. This is a special case of heterogeneous DA, with the existence of both common features and domain specific features. We name this kind of problems as *hybrid domain adaptation* problem. To the best of our knowledge, this is the first work of studying such a problem setting, in spite of its popularity in real applications.

In hybrid DA, the main challenge lies in the heterogeneity of domain specific features. Existing heterogeneous DA methods cannot be applied because of the lack of supplementary information. Alternatively, we could convert the hybrid problem to a homogeneous one. Two different strategies can be used to do so: “*cut-off*” and “*fill-up*”. *Cut-off* is a straightforward way to align the feature spaces by discarding the domain specific features and retaining only the common features. *Fill-up* is another alternative to eliminate the feature heterogeneities. The idea is to expand the feature space by concatenating common features and specific features of the two domains, after which the feature spaces become homogeneous.

Figure 1. A sentiment analysis example

In general, the *fill-up* strategy is preferred to the *cut-off* one. This is because *fill-up* preserves domain specific features, which typically encode the discriminative information and are thus critical for the learning task. Such information is particularly essential in unsupervised DA, in which the target domain has no labeled data to guide the learning. The current common way of doing *fill-up* is filling in zeros to those specific features appearing in one domain but not the other, called *F_Zero*. In the text field, for instance, the vocabulary lexicons of different domains are often merged, with the words specific to one domain filled up with zeros for other domains [2], [5], [10], [16], [17], [18]. However, our analyses demonstrate that *F_Zero* does not benefit from using domain specific features in adaptation, and it is actually equivalent to the “*cut-off*” strategy that only uses the common features. We show the evidence and discuss more details in section 5.4.

In this paper, we propose a general framework called “Domain Specific Feature Transfer” (DSFT) for hybrid DA, which can effectively utilize the domain specific features. DSFT takes the *fill-up* strategy in a way that it benefits domain adaptation. The *fill-up* in the two different domains is bridged by the common features, and meanwhile the discriminative power embedded in the domain specific features is captured and transferred from one domain to the other. More specifically, we learn a feature mapping between common features and domain specific features in each domain. We then apply the learned mappings across domains to do the *fill-up*. During the learning, we enforce the distribution discrepancy between the two domains after the *fill-up* to be minimized. By doing so, the *fill-up* in DSFT not only transfers the knowledge across domains, but also brings the two domains closer for better adaptation. The learning of the mappings, the *fill-up*, as well as the minimization of the distribution discrepancy, are incorporated into one unified framework. With the learnt mappings, an expanded homogeneous feature space is constructed, and DSFT then trains a model in the source domain and makes predictions in the target domain in this new space. In this paper, we propose two instantiations of DSFT by considering both the linear and nonlinear cases. A closed-form solution to each case is derived.

We provide a real-world example, shown in Figure 1, to further help clarify our DSFT. The task is to transfer sentiment knowledge between reviews of different products, e.g., books and electronic products. Both sets of reviews share some common sentiment words, e.g., “good”, “poor”, etc. Each product also has its own specific sentiment words, e.g., “comprehensive” for books, “noisy” for electronic products, etc. DSFT learns the relationship between the common words and the specific words for each product. For instance, a linear relationship gives $\text{'comprehensive'} = 0.8 \times \text{'great'} + 0.6 \times \text{'good'} + 0.1 \times \text{'work'} - 0.2 \times \text{'poor'}$. Then, DSFT transfers the learned relationships across domains. Adapted books specific sentiment words, e.g., ‘comprehensive’ and ‘superficial’, appear in the reviews of electronic products. These adapted books specific sentiment words encode rich sentiment information, and thus may benefit for the sentiment analysis of electronic products. Consequently, a universal homogeneous feature space for both products is constructed. DSFT further aligns the two products by minimizing the distribution discrepancies of the universal homogeneous feature space.

Though the idea of DSFT may be simple, its underlying mechanism is very effective. First, the *fill-up* by the learnt mappings is essentially to reweight the common features by exploiting the domain specific features. By reweighting, the discriminative power embedded in the domain specific features is brought over to the common features, and the two domains are brought to be closer. Second, the *fill-up* and the minimization of distribution discrepancy enable DSFT to provide a better feature representation for adaptation. The constructed universal features have smaller domain divergence and source training errors. These two points are empirically verified in our experiments.

The main contributions of this paper are: (1) we propose a new hybrid DA problem; (2) we develop a novel general framework called DSFT for hybrid DA without using supplementary information, and instantiate a linear case and a nonlinear case; (3) we conduct extensive experiments and demonstrate the superiority of DSFT to state-of-the-art DA methods with “*cut-off*” and “*fill-up*” strategies.

The rest of the paper is organized as follows. Section 2 gives related works on domain adaptation methods and Maximum Mean Discrepancy (MMD), the metric we utilize to measure the distribution discrepancy. Section 3 describes our proposed general framework DSFT. Section 4 instantiate a linear case and a nonlinear case for DSFT. Section 5 presents the empirical study and comparison. Finally, section 6 concludes the paper.

2 RELATED WORK

Homogeneous DA aims to adapt learning models across domains with the same feature space. The main idea is to reduce the distribution discrepancies through appropriate feature mappings. The structural correspondence learning methods [5], [16] sought to learn a shared feature representation by utilizing pivot features. In [19], [20], [21], PCA-based dimensionality reduction methods were investigated to find a common latent space shared by the two domains. In [22], an FEDA method was proposed, which constructs an expanded feature space by replicating the features three times and filling up with zeros for some of the replicates. In [7], geodesic flow kernel (GFK) was proposed to generate the intermediate subspaces to model the domain shift. To address the computational issue of GFK, Fernando et al. [23] proposed a subspace alignment method to align the source and target

Figure 2. Framework of DSFT

subspaces directly. In [24], a transfer kernel learning method was proposed to learn a domain invariant kernel between domains. Clinchant et al. [25] presented a target regularization for transfer.

Heterogeneous DA aims to transfer knowledge across domains with different feature spaces. Since there is no correspondence between feature spaces, supplementary information is involved to link up domains. In [26], [27] and [13], source and target labels are used to align the feature spaces. Instances with the same labels are mapped to be closer, while instances with different labels are mapped to be far away from each other. Dai et al. [14] proposed a translator to obtain the correspondence between two domains. In [15], correspondence instances were used to learn the relationship between the source and target domains. In [28], Li et al. proposed a heterogeneous feature augmentation method. Target labels are used to connect the two domains in the augmented space.

The hybrid DA studied in this paper is a new problem setting. Compared to the well-studied homogeneous DA and heterogeneous DA, we emphasize the following two points. First, since it is straightforward to convert hybrid DA to homogeneous DA by the “cut-off” and “fill-up” strategies, we compare our DSFT proposed for hybrid DA with an important class of baselines that apply the homogeneous DA methods after the “cut-off” or “fill-up”. Second, although hybrid DA can be treated as a special case of heterogeneous DA, the existence of common features enables it to be solved without supplementary information, which makes it incomparable to existing heterogeneous methods.

To quantify the distribution discrepancy, we adopt a metric called Maximum Mean Discrepancy (MMD) [29]. It does not assume any parametric form for the distributions and thus is widely used to measure the distance between two distributions in domain adaptation [21], [30], [31]. It takes the mean embeddings of the two distributions in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ as a distance calculation to avoid the density estimation. Given two sets of samples $\mathbf{U} = \{\mathbf{u}_1, \dots, \mathbf{u}_{n_1}\}$ and $\mathbf{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_{n_2}\}$, drawn from the distributions P and Q respectively, the empirical MMD is calculated as:

$$\left\| \frac{1}{n_1} \sum_{i=1}^{n_1} \Phi(\mathbf{u}_i) - \frac{1}{n_2} \sum_{i=1}^{n_2} \Phi(\mathbf{v}_i) \right\|_{\mathcal{H}},$$

where $\Phi: \mathbb{R} \rightarrow \mathcal{H}$ is the feature mapping.

3 THE GENERAL DSFT FRAMEWORK

In this section, we introduce the problem setting of hybrid DA, and then present a general DSFT framework for this problem setting.

3.1 Problem Statement

In a hybrid DA problem, we are given labeled data in the source domain and unlabeled data in the target domain. The two domains share a set of common input features, while each of them has its own domain specific input features. Let $\mathbf{X}_s = [\mathbf{S}_c; \mathbf{S}_s] \in \mathbb{R}^{(c+s) \times n_s}$ denote the source data matrix with n_s samples and $\mathbf{X}_t = [\mathbf{T}_c; \mathbf{T}_t] \in \mathbb{R}^{(c+t) \times n_t}$ denote the target data matrix with n_t samples. Herein, $\mathbf{S}_c \in \mathbb{R}^{c \times n_s}$ and $\mathbf{T}_c \in \mathbb{R}^{c \times n_t}$ are the feature matrices with c common features, $\mathbf{S}_s \in \mathbb{R}^{s \times n_s}$ is the feature matrix of s source specific features, and $\mathbf{T}_t \in \mathbb{R}^{t \times n_t}$ is the feature matrix with t target specific features. The labels of $\mathbf{X}_s$, denoted as $\mathbf{Y}_s$, are available. Our task is to predict the labels $\mathbf{Y}_t$ in the target domain, by leveraging the information from the source domain.

3.2 Domain Specific Feature Transfer

In this section, we describe how the domain specific features are utilized to facilitate the domain adaptation.

Figure 2 shows the framework of our DSFT. The *fill-up* and adaptation is accomplished by using the common features in the two domains as the bridge. Specifically, we learn feature mappings Ψ_s and Ψ_t as the translations to link up the common features with the domain specific features, respectively for each domain. The objective functions are:

$$\min_{\Psi_s} \|\Psi_s(\mathbf{S}_c) - \mathbf{S}_s\|_F^2, \quad \min_{\Psi_t} \|\Psi_t(\mathbf{T}_c) - \mathbf{T}_t\|_F^2. \quad (1)$$

After obtaining the translations, we cross-use them to do the *fill-up*. Precisely, we apply the feature mapping Ψ_t (i.e., the translation obtained from the target domain) to the common features of the source domain. This gives a matrix that encodes the target-specific feature expressions for the source samples by using the common-specific rule in the target domain. We denote this matrix as $\mathbf{S}_a \in \mathbb{R}^{t \times n_s}$, the adapted target-specific feature matrix. We do the corresponding translation for the target domain and obtain the adapted matrix $\mathbf{T}_a \in \mathbb{R}^{s \times n_t}$. The computation of $\mathbf{S}_a$ and $\mathbf{T}_a$ is given below:

$$\mathbf{S}_a = \Psi_t(\mathbf{S}_c), \quad \mathbf{T}_a = \Psi_s(\mathbf{T}_c). \quad (2)$$

We then utilize both the original features and the *fill-up* adapted features for adaptation. This gives a homogeneous feature space as follows:

$$\mathbf{X}_{s_h} = [\mathbf{S}_c; \mathbf{S}_s; \mathbf{S}_a], \quad \mathbf{X}_{t_h} = [\mathbf{T}_c; \mathbf{T}_a; \mathbf{T}_t]. \quad (3)$$

Algorithm 1 Domain Specific Feature Transfer

Input: Source data matrix $\mathbf{X}_s$, source label $\mathbf{Y}_s$, target data matrix $\mathbf{X}_t$
Procedure:
 1. Compute Ψ_s and Ψ_t by solving objective function (5);
 2. Compute $\mathbf{X}_{s_h}$ and $\mathbf{X}_{t_h}$ by Eq. (3)
 3. Train a model $\mathcal{M}$ from $\mathbf{X}_{s_h}$ and $\mathbf{Y}_s$;
 4. Predict $\mathbf{Y}_t$ for $\mathbf{X}_{t_h}$ using $\mathcal{M}$.
Output: $\mathbf{Y}_t$.

3.3 Distribution Discrepancy Minimization

So far the translation between two domains is performed individually for each domain, as indicated in Eq. (1). However, our objective is to adapt the source domain to the target domain so that it helps the prediction task in the latter. For this purpose, we enforce the marginal distributions of the two domains to be as close as possible, i.e., $\mathbf{X}_{s_h}$ and $\mathbf{X}_{t_h}$, to have a low distribution discrepancy. More specifically, we aim to choose among different possible translations those with better adaptation capability.

To do so, we propose to quantify the distribution discrepancy between $\mathbf{X}_{s_h}$ and $\mathbf{X}_{t_h}$ with a divergence metric $Dist(\mathbf{X}_{s_h}, \mathbf{X}_{t_h})$, and incorporate it into the optimization. To minimize the distribution discrepancy, we need to minimize $Dist(\mathbf{X}_{s_h}, \mathbf{X}_{t_h})$ in the learning of feature mappings:

$$\min_{\Psi_s, \Psi_t} Dist(\mathbf{X}_{s_h}, \mathbf{X}_{t_h}). \quad (4)$$

3.4 Final DSFT framework

By combining (1) and (4), we obtain the final objective function of DSFT with three terms:

$$\begin{aligned} & \min_{\Psi_s, \Psi_t} \underbrace{\|\Psi_s(\mathbf{S}_c) - \mathbf{S}_s\|_F^2 + \|\Psi_t(\mathbf{T}_c) - \mathbf{T}_t\|_F^2}_{\text{Translation}} \\ & + \underbrace{\alpha Dist(\mathbf{X}_{s_h}, \mathbf{X}_{t_h})}_{\text{Discrepancy}} + \underbrace{\beta(\|\Psi_s\|_F^2 + \|\Psi_t\|_F^2)}_{\text{Regularization}}, \end{aligned} \quad (5)$$

where the regularization term is used to control the complexity of the mappings, and α and β are hyper-parameters.

After obtaining the optimal Ψ_s and Ψ_t by solving (5), we can compute the data matrices $\mathbf{X}_{s_h}$ and $\mathbf{X}_{t_h}$ by Eq. (3). We then train a prediction model from $\mathbf{X}_{s_h}$ and $\mathbf{Y}_s$ in the source domain, and apply it to predict the labels $\mathbf{Y}_t$ for $\mathbf{X}_{t_h}$ in the target domain. Our DSFT framework is applicable for both classification and regression tasks. Existing classification/regression methods can be used for training the models. For out-of-sample target data, we can use Ψ_s to do the *fill-up* and form the homogenous feature space. The prediction can then be done by the trained model. The proposed DSFT is summarized in Algorithm 1.

The proposed DSFT framework is general in the sense that it is not tailored to the specific types of feature mappings and distribution discrepancy metrics. In the following sections, we instantiate the framework to a linear case and a nonlinear case by specifying two particular mappings with a popular distribution discrepancy metric.

4 INSTANTIATION OF DSFT FRAMEWORK

To instantiate the DSFT framework, we need to specify the feature mappings and the distribution discrepancy metric in the objective function (5). In this section, we first introduce a nonparametric

metric to measure the distribution discrepancy, and then propose a linear instantiation DSFT_l and a nonlinear instantiation DSFT_{nl}.

4.1 A Distribution Discrepancy Metric MMD

In this work, we propose to use Maximum Mean Discrepancy (MMD) [29] as a quantifying metric to measure the distribution discrepancy. It is commonly used to measure the distance between two distributions [21] since it avoids density estimations. In our context, the empirical MMD between $\mathbf{X}_{s_h}$ and $\mathbf{X}_{t_h}$ is $(\mathbf{x}_{s_h_i}$ and $\mathbf{x}_{t_h_j}$ is the single instance in the source and target domains, respectively):

$$\begin{aligned} Dist(\mathbf{X}_{s_h}, \mathbf{X}_{t_h}) &= \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{x}_{s_h_i} - \frac{1}{n_t} \sum_{j=1}^{n_t} \mathbf{x}_{t_h_j} \right\|^2 \\ &= tr(\mathbf{X} \mathbf{M} \mathbf{X}^T), \end{aligned} \quad (6)$$

where tr is the the operator to calculate the trace of a matrix, $\mathbf{X} = [\mathbf{X}_{s_h}, \mathbf{X}_{t_h}] \in \mathbb{R}^{(c+s+t) \times (n_s+n_t)}$, and $\mathbf{M}$ is defined as follows:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11}, \mathbf{M}_{12} \\ \mathbf{M}_{21}, \mathbf{M}_{22} \end{bmatrix}$$

with $\mathbf{M}_{11} = \frac{1}{n_s^2} \mathbf{1}_{n_s \times n_s}$, $\mathbf{M}_{12} = \mathbf{M}_{21}^T = -\frac{1}{n_t n_s} \mathbf{1}_{n_s \times n_t}$ and $\mathbf{M}_{22} = \frac{1}{n_t^2} \mathbf{1}_{n_t \times n_t}$ ($\mathbf{1}_{p \times q}$ is an $p \times q$ matrix with all ones).

4.2 A Linear Case DSFT_l

In this section, we instantiate a linear case of DSFT, DSFT_l, by simply assuming that the mappings between the common features and the domain specific features are linear transformations. Specifically, we represent the two feature mappings using two matrices $\mathbf{W}_s$ and $\mathbf{W}_t$, which means

$$\Psi_s(\mathbf{S}_c) = \mathbf{W}_s^T \mathbf{S}_c, \quad \Psi_t(\mathbf{T}_c) = \mathbf{W}_t^T \mathbf{T}_c. \quad (7)$$

By applying Eqs. (7) to Eqs. (2), we can construct the new homogeneous feature space as follows:

$$\mathbf{X}_{s_h}^l = [\mathbf{S}_c; \mathbf{S}_s; \mathbf{S}_a^l], \quad \mathbf{X}_{t_h}^l = [\mathbf{T}_c; \mathbf{T}_a^l; \mathbf{T}_t], \quad (8)$$

where $\mathbf{S}_a^l = \mathbf{W}_t^T \mathbf{S}_c$ and $\mathbf{T}_a^l = \mathbf{W}_s^T \mathbf{T}_c$. Then we plug $\mathbf{X}_{s_h}^l$ and $\mathbf{X}_{t_h}^l$ of Eq. (8) into Eq. (6), and further derive:

$$\begin{aligned} Dist_l(\mathbf{X}_{s_h}^l, \mathbf{X}_{t_h}^l) &= tr([\mathbf{X}_{s_h}^l, \mathbf{X}_{t_h}^l] \mathbf{M} [\mathbf{X}_{s_h}^l, \mathbf{X}_{t_h}^l]^T) \\ &= tr([\mathbf{S}_c, \mathbf{T}_c] \mathbf{M} [\mathbf{S}_c, \mathbf{T}_c]^T) + tr([\mathbf{S}_s, \mathbf{T}_a^l] \mathbf{M} [\mathbf{S}_s, \mathbf{T}_a^l]^T) \\ &\quad + tr([\mathbf{S}_a^l, \mathbf{T}_t] \mathbf{M} [\mathbf{S}_a^l, \mathbf{T}_t]^T) \\ &= tr(\mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^T + 2 \mathbf{S}_c \mathbf{M}_{12} \mathbf{T}_c^T + \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^T) \\ &\quad + tr(\mathbf{S}_s \mathbf{M}_{11} \mathbf{S}_s^T + 2 \mathbf{W}_s^T \mathbf{T}_c \mathbf{M}_{21} \mathbf{S}_s^T + \mathbf{W}_s^T \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^T \mathbf{W}_s) \\ &\quad + tr(\mathbf{T}_t \mathbf{M}_{22} \mathbf{T}_t^T + 2 \mathbf{W}_t^T \mathbf{S}_c \mathbf{M}_{12} \mathbf{T}_t^T + \mathbf{W}_t^T \mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^T \mathbf{W}_t). \end{aligned} \quad (9)$$

Finally, we can obtain the objective function of DSFT_l as follows:

$$\begin{aligned} & \min_{\mathbf{W}_s, \mathbf{W}_t} \left\| \mathbf{W}_s^T \mathbf{S}_c - \mathbf{S}_s \right\|_F^2 + \left\| \mathbf{W}_t^T \mathbf{T}_c - \mathbf{T}_t \right\|_F^2 \\ & + \alpha Dist_l(\mathbf{X}_{s_h}^l, \mathbf{X}_{t_h}^l) + \beta(\|\mathbf{W}_s\|_F^2 + \|\mathbf{W}_t\|_F^2), \end{aligned} \quad (10)$$

To solve the optimization problem (10), we first denote the final objective function as L , and define $L = L_1 + L_2 + L_3$ where

$$\begin{aligned} L_1 &= \left\| \mathbf{W}_s^\top \mathbf{S}_c - \mathbf{S}_s \right\|_F^2 + \beta \left\| \mathbf{W}_s \right\|_F^2 \\ &= \text{tr}(\mathbf{S}_c^\top \mathbf{W}_s \mathbf{W}_s^\top \mathbf{S}_c - 2\mathbf{S}_c^\top \mathbf{W}_s \mathbf{S}_s + \mathbf{S}_s^\top \mathbf{S}_s) \\ &\quad + \beta \text{tr}(\mathbf{W}_s^\top \mathbf{W}_s), \\ L_2 &= \left\| \mathbf{W}_t^\top \mathbf{T}_c - \mathbf{T}_t \right\|_F^2 + \beta \left\| \mathbf{W}_t \right\|_F^2 \\ &= \text{tr}(\mathbf{T}_c^\top \mathbf{W}_t \mathbf{W}_t^\top \mathbf{T}_c - 2\mathbf{T}_c^\top \mathbf{W}_t \mathbf{T}_t + \mathbf{T}_t^\top \mathbf{T}_t) \\ &\quad + \beta \text{tr}(\mathbf{W}_t^\top \mathbf{W}_t), \\ L_3 &= \alpha \text{Dist}_l(\mathbf{X}_{s_h}^l, \mathbf{X}_{t_h}^l). \end{aligned}$$

According to Eq. (9), L_3 is the summation of three trace terms. These three trace terms correspond to the discrepancy term of the common features, the source specific features, and the target specific features, respectively. Note that none of these three trace terms contain $\mathbf{W}_s$ and $\mathbf{W}_t$ simultaneously. This is a good property which leads to a closed-form solution for the final objective function. Specifically, we calculate the first-order derivatives of L with respect to $\mathbf{W}_s$ and $\mathbf{W}_t$:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{W}_s} &= \frac{\partial L_1}{\partial \mathbf{W}_s} + \frac{\partial L_3}{\partial \mathbf{W}_s}, \\ \frac{\partial L}{\partial \mathbf{W}_t} &= \frac{\partial L_2}{\partial \mathbf{W}_t} + \frac{\partial L_3}{\partial \mathbf{W}_t}, \end{aligned} \quad (11)$$

where the first-order derivatives with respect to $\mathbf{W}_s$ and $\mathbf{W}_t$ for L_1 , L_2 and L_3 are:

$$\begin{aligned} \frac{\partial L_1}{\partial \mathbf{W}_s} &= 2\mathbf{W}_s^\top \mathbf{S}_c \mathbf{S}_c^\top - 2\mathbf{S}_s \mathbf{S}_c^\top + 2\beta \mathbf{W}_s^\top, \\ \frac{\partial L_3}{\partial \mathbf{W}_s} &= 2\alpha \mathbf{W}_s^\top \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top + 2\alpha \mathbf{S}_s \mathbf{M}_{12} \mathbf{T}_c^\top, \\ \frac{\partial L_2}{\partial \mathbf{W}_t} &= 2\mathbf{W}_t^\top \mathbf{T}_c \mathbf{T}_c^\top - 2\mathbf{T}_t \mathbf{T}_c^\top + 2\beta \mathbf{W}_t^\top, \\ \frac{\partial L_3}{\partial \mathbf{W}_t} &= 2\alpha \mathbf{W}_t^\top \mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + 2\alpha \mathbf{T}_t \mathbf{M}_{21} \mathbf{S}_c^\top. \end{aligned}$$

By setting Eqs. (11) to zero, we obtain the optimal $\mathbf{W}_s$ and $\mathbf{W}_t$ as follows:

$$\begin{aligned} \mathbf{W}_s &= (\mathbf{A}\mathbf{B}^{-1})^\top, \\ \mathbf{A} &= \mathbf{S}_s \mathbf{S}_c^\top - \alpha \mathbf{S}_s \mathbf{M}_{12} \mathbf{T}_c^\top, \\ \mathbf{B} &= \mathbf{S}_c \mathbf{S}_c^\top + \alpha \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top + \beta \mathbf{I}, \\ \mathbf{W}_t &= (\mathbf{C}\mathbf{D}^{-1})^\top, \\ \mathbf{C} &= \mathbf{T}_t \mathbf{T}_c^\top - \alpha \mathbf{T}_t \mathbf{M}_{21} \mathbf{S}_c^\top, \\ \mathbf{D} &= \mathbf{T}_c \mathbf{T}_c^\top + \alpha \mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + \beta \mathbf{I}. \end{aligned} \quad (12)$$

4.3 A Nonlinear Case DSFT_{nl}

In DSFT_l, a simple linear relationship is assumed between the common features and the domain specific features. However, in the real-world data, the relationship between the common features and the domain specific features is usually too diverse and complex to be modeled by a simple linear transformation. This motivates us to explore the more complicated nonlinear relationship between the two sets of features. Moreover, exploring nonlinearity also brings benefits to the distribution discrepancies minimization by MMD as it enables nonlinear mappings from $\mathbb{R} \rightarrow \mathcal{H}$.

In this section, we instantiate a nonlinear case of DSFT, DSFT_{nl}, by assuming a nonlinear relationship between the common features and the domain specific features. To take the nonlinearity across the features into account, we propose a kernelized solution. Specifically, we first use a nonlinear mapping Φ to map the source and target common features to RKHS, respectively, where $\Phi : \mathbb{R}^c \rightarrow \mathcal{H}$ with $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x}_j)$ as the kernel of $\mathcal{H}$. Then we construct the specific features by the mapped common features in $\mathcal{H}$ using a linear transformation for each domain. Thus, we have the nonlinear mappings¹:

$$\Psi_s(\mathbf{S}_c) = \mathbf{Q}_s^\top \Phi(\mathbf{S}_c), \quad \Psi_t(\mathbf{T}_c) = \mathbf{Q}_t^\top \Phi(\mathbf{T}_c). \quad (13)$$

Correspondingly, we can obtain the nonlinear version of Eqs. (1) and (2):

$$\begin{aligned} \min_{\mathbf{Q}_s} \left\| \mathbf{Q}_s^\top \Phi(\mathbf{S}_c) - \mathbf{S}_s \right\|_F^2, \quad \min_{\mathbf{Q}_t} \left\| \mathbf{Q}_t^\top \Phi(\mathbf{T}_c) - \mathbf{T}_t \right\|_F^2. \\ \mathbf{S}_a^{nl} = \mathbf{Q}_t^\top \Phi(\mathbf{S}_c), \quad \mathbf{T}_a^{nl} = \mathbf{Q}_s^\top \Phi(\mathbf{T}_c), \end{aligned}$$

According to [32], [33], the linear mapping in $\mathcal{H}$ can be represented as a linear combination of the data points in $\mathcal{H}$, that is, for each domain we have $\mathbf{Q}_s^\top = \mathbf{Q}_{sk}^\top \Phi(\mathbf{S}_c)^\top$ and $\mathbf{Q}_t^\top = \mathbf{Q}_{tk}^\top \Phi(\mathbf{T}_c)^\top$. By denoting $\mathbf{K}_{ss} = \Phi(\mathbf{S}_c)^\top \Phi(\mathbf{S}_c)$, $\mathbf{K}_{tt} = \Phi(\mathbf{T}_c)^\top \Phi(\mathbf{T}_c)$, $\mathbf{K}_{st} = \Phi(\mathbf{S}_c)^\top \Phi(\mathbf{T}_c)$, and $\mathbf{K}_{ts} = \mathbf{K}_{st}^\top$, we obtain the kernelized version of Eqs. (1) and (2):

$$\begin{aligned} \min_{\mathbf{Q}_{sk}} \left\| \mathbf{Q}_{sk}^\top \mathbf{K}_{ss} - \mathbf{S}_s \right\|_F^2, \quad \min_{\mathbf{Q}_{tk}} \left\| \mathbf{Q}_{tk}^\top \mathbf{K}_{tt} - \mathbf{T}_t \right\|_F^2, \\ \mathbf{S}_a^{nl} = \mathbf{Q}_{tk}^\top \mathbf{K}_{ts}, \quad \mathbf{T}_a^{nl} = \mathbf{Q}_{sk}^\top \mathbf{K}_{st}. \end{aligned} \quad (14)$$

Then we construct the new $\mathbf{X}_{s_h}^{nl}$ and $\mathbf{X}_{t_h}^{nl}$ using Eq. (3), and derive the new discrepancy term as:

$$\begin{aligned} \text{Dist}_{nl}(\mathbf{X}_{s_h}^{nl}, \mathbf{X}_{t_h}^{nl}) \\ = \text{tr}(\mathbf{S}_c \mathbf{M}_{11} \mathbf{S}_c^\top + 2\mathbf{S}_c \mathbf{M}_{12} \mathbf{T}_c^\top + \mathbf{T}_c \mathbf{M}_{22} \mathbf{T}_c^\top) \\ + \text{tr}(\mathbf{S}_s \mathbf{M}_{11} \mathbf{S}_s^\top + 2\mathbf{Q}_{sk}^\top \mathbf{K}_{st} \mathbf{M}_{21} \mathbf{S}_s^\top \\ + \mathbf{Q}_{sk}^\top \mathbf{K}_{st} \mathbf{M}_{22} \mathbf{K}_{st}^\top \mathbf{Q}_{sk}) \\ + \text{tr}(\mathbf{T}_t \mathbf{M}_{22} \mathbf{T}_t^\top + 2\mathbf{Q}_{tk}^\top \mathbf{K}_{ts} \mathbf{M}_{12} \mathbf{T}_t^\top \\ + \mathbf{Q}_{tk}^\top \mathbf{K}_{ts} \mathbf{M}_{11} \mathbf{K}_{ts}^\top \mathbf{Q}_{tk}). \end{aligned} \quad (15)$$

Combining (14) and (15), we obtain the kernelized final objective function of DSFT_{nl}:

$$\begin{aligned} \min_{\mathbf{Q}_{sk}, \mathbf{Q}_{tk}} \left\| \mathbf{Q}_{sk}^\top \mathbf{K}_{ss} - \mathbf{S}_s \right\|_F^2 + \left\| \mathbf{Q}_{tk}^\top \mathbf{K}_{tt} - \mathbf{T}_t \right\|_F^2 \\ + \alpha \text{Dist}_{nl}(\mathbf{X}_{s_h}^{nl}, \mathbf{X}_{t_h}^{nl}) + \beta (\left\| \mathbf{Q}_{sk} \right\|_F^2 + \left\| \mathbf{Q}_{tk} \right\|_F^2). \end{aligned} \quad (16)$$

Similar to the linear case, we can obtain the optimal closed-form solutions $\mathbf{Q}_{sk}$ and $\mathbf{Q}_{tk}$ of (16) as follows:

$$\begin{aligned} \mathbf{Q}_{sk} &= (\mathbf{A}_k \mathbf{B}_k^{-1})^\top, \\ \mathbf{A}_k &= \mathbf{S}_s \mathbf{K}_{ss}^\top - \alpha \mathbf{S}_s \mathbf{M}_{12} \mathbf{K}_{st}^\top, \\ \mathbf{B}_k &= \mathbf{K}_{ss} \mathbf{K}_{ss}^\top + \alpha \mathbf{K}_{st} \mathbf{M}_{22} \mathbf{K}_{st}^\top + \beta \mathbf{I}, \\ \mathbf{Q}_{tk} &= (\mathbf{C}_k \mathbf{D}_k^{-1})^\top, \\ \mathbf{C}_k &= \mathbf{T}_t \mathbf{K}_{tt}^\top - \alpha \mathbf{T}_t \mathbf{M}_{21} \mathbf{K}_{ts}^\top, \\ \mathbf{D}_k &= \mathbf{K}_{tt} \mathbf{K}_{tt}^\top + \alpha \mathbf{K}_{ts} \mathbf{M}_{11} \mathbf{K}_{ts}^\top + \beta \mathbf{I}. \end{aligned} \quad (17)$$

1. Note that there are other ways to instantiate the nonlinear relationships. By formulating Eq. (13), we can obtain a closed-form solution for the nonlinear case, which is also similar to the linear one.

4.4 Complexity Analysis on DSFT_l and DSFT_{nl}

Both DSFT_l and DSFT_{nl} are very effective to be solved as they have closed-form solutions. The closed form solutions can be obtained by solving systems of linear equations as shown in Eqs. (12) and Eqs. (17). The computational time complexity in solving these systems of linear equations is dominated by the calculation of matrix inverse. Specifically, the computational time complexity of DSFT_l and DSFT_{nl} is $\mathcal{O}(c^3)$ and $\mathcal{O}(n^3)$ with $n = \max(n_s, n_t)$, respectively. When the number of common features is much less than the number of samples in each domain, DSFT_l is computational cheaper than DSFT_{nl}. However, when the common feature space of two domains is huge and sparse, DSFT_{nl} may be more efficient.

5 EXPERIMENTAL STUDY

In this section, we firstly investigate DSFT_l and DSFT_{nl} with different nonlinear kernels on synthetic datasets. We then compare our DSFT, i.e. DSFT_l and DSFT_{nl}, with state-of-the-art DA methods on two real-world datasets. Finally, we analyze the properties of our DSFT.

5.1 Synthetic Datasets & Experiments

In this section, we aim to (1) demonstrate the effectiveness of the proposed DSFT, and (2) investigate the performance of DSFT_{nl} with different nonlinear kernels. We conduct experiments on synthetic Datasets.

Synthetic Datasets. We consider to generate data for a linear regression problem. Firstly, we generate 100 input data following the hybrid DA setting for the source and target domains, respectively. We utilize the “normrnd” function in Matlab R2014a to generate common features. Specifically, source common features are generated from a normal distribution with mean as 0 and standard deviation as 1. Target common features are generated from a normal distribution with mean as 0.5 and standard deviation as 0.5. Thus, we obtain $\mathbf{S}_c \in \mathbb{R}^{100 \times 10}$ and $\mathbf{T}_c \in \mathbb{R}^{100 \times 10}$. We then randomly generate two mappings $\mathbf{W}_s \in \mathbb{R}^{10 \times 6}$ and $\mathbf{W}_t \in \mathbb{R}^{10 \times 8}$. With these two mappings, we generate domain specific features for the two domains. For the linear relationship case, we generate $\mathbf{S}_s^l \in \mathbb{R}^{100 \times 6}$ using $\mathbf{S}_c \mathbf{W}_s$, and $\mathbf{T}_t^l \in \mathbb{R}^{100 \times 8}$ using $\mathbf{T}_c \mathbf{W}_t$. To insert nonlinearity, we apply the “tansig” function in Matlab R2014a to $\mathbf{S}_s^l$ and $\mathbf{T}_t^l$, and thus we obtain $\mathbf{S}_s^{nl}$ and $\mathbf{T}_t^{nl}$ for the nonlinear relationship case. By doing so, we obtain two sets of input data, namely $[\mathbf{S}_c, \mathbf{S}_s^l] \in \mathbb{R}^{100 \times 16}$ with $[\mathbf{T}_c, \mathbf{T}_t^l] \in \mathbb{R}^{100 \times 18}$ for the linear relationship case, and $[\mathbf{S}_c, \mathbf{S}_s^{nl}] \in \mathbb{R}^{100 \times 16}$ with $[\mathbf{T}_c, \mathbf{T}_t^{nl}] \in \mathbb{R}^{100 \times 18}$ for the nonlinear relationship case. Secondly, we generate function values for these input data. We use $f_s(\mathbf{x}) = \omega_s \mathbf{x} + \epsilon$, where $\omega_s \in \mathbb{R}^{16}$ is a randomly generated weights vector and ϵ is a zero-mean gaussian noise term, as the source output function. Likewise, we use $f_t(\mathbf{x}) = \omega_t \mathbf{x} + \epsilon$, where $\omega_t \in \mathbb{R}^{18}$, as the target output function. We apply these two output functions to the two sets of input data, and obtain $\mathbf{y}_s^l$ with $\mathbf{y}_t^l$ for the linear relationship case, and $\mathbf{y}_s^{nl}$ with $\mathbf{y}_t^{nl}$ for the nonlinear relationship case. In sum, with one execution of the above data generation steps, we can construct two datasets $\mathbf{D}^l = \{[\mathbf{S}_c, \mathbf{S}_s^l], [\mathbf{T}_c, \mathbf{T}_t^l]\}$, in which the relationship between common features and domain specific features is linear, and $\mathbf{D}^{nl} = \{[\mathbf{S}_c, \mathbf{S}_s^{nl}], [\mathbf{T}_c, \mathbf{T}_t^{nl}]\}$, in which the relationship between common features and domain specific features is nonlinear.

Table 1
Comparison results on $\mathbf{D}_i^l, i = 1, \dots, 15$

Datasets	COM	DSFT _l	DSFT _{poly}	DSFT _{acos}	DSFT _{rbf}
$\mathbf{D}_1^l$	2.0510	0.8556	0.9991	0.8134	0.4814
$\mathbf{D}_2^l$	1.3862	0.9855	0.9600	1.0013	0.9811
$\mathbf{D}_3^l$	0.8336	0.4362	0.4011	0.4274	0.3312
$\mathbf{D}_4^l$	0.9039	0.5893	0.3922	0.5883	0.2752
$\mathbf{D}_5^l$	0.4484	0.1474	0.2254	0.0998	0.1018
$\mathbf{D}_6^l$	1.3044	0.6760	0.5730	0.6024	0.6318
$\mathbf{D}_7^l$	0.4421	0.3494	0.2352	0.1805	0.1465
$\mathbf{D}_8^l$	1.3897	1.0190	1.0412	0.6657	0.2636
$\mathbf{D}_9^l$	2.9802	0.9544	1.3151	0.9681	0.8594
$\mathbf{D}_{10}^l$	0.8390	0.2939	0.3017	0.2034	0.2311
$\mathbf{D}_{11}^l$	1.1824	0.5698	0.6694	0.6539	0.8799
$\mathbf{D}_{12}^l$	1.8538	0.9212	0.9239	0.5023	0.1529
$\mathbf{D}_{13}^l$	0.9381	0.3546	0.3269	0.3485	0.3769
$\mathbf{D}_{14}^l$	0.5334	0.3845	0.3368	0.2077	0.2513
$\mathbf{D}_{15}^l$	0.5974	0.1131	0.1314	0.1601	0.2480
Mean	1.1789	0.5773	0.5888	0.4950	0.4141

Table 2
Comparison results on $\mathbf{D}_i^{nl}, i = 1, \dots, 15$

Datasets	COM	DSFT _l	DSFT _{poly}	DSFT _{acos}	DSFT _{rbf}
$\mathbf{D}_1^{nl}$	3.3510	1.9514	2.3107	2.1545	1.5772
$\mathbf{D}_2^{nl}$	4.2997	2.8531	3.0380	2.9961	2.6450
$\mathbf{D}_3^{nl}$	1.4887	1.1587	1.1078	1.1511	1.0946
$\mathbf{D}_4^{nl}$	1.4764	1.3887	0.9656	1.5851	1.0740
$\mathbf{D}_5^{nl}$	0.8857	0.5561	0.5638	0.3869	0.3894
$\mathbf{D}_6^{nl}$	4.1302	2.3168	2.4579	2.4010	2.2626
$\mathbf{D}_7^{nl}$	1.1443	0.8613	0.6801	0.6039	0.5434
$\mathbf{D}_8^{nl}$	1.9772	1.7385	1.4402	1.6624	0.7497
$\mathbf{D}_9^{nl}$	2.6205	2.0275	2.1034	1.5632	1.7557
$\mathbf{D}_{10}^{nl}$	1.0525	0.7274	0.5724	0.4833	0.5516
$\mathbf{D}_{11}^{nl}$	2.9116	2.3356	1.9944	1.7465	2.3631
$\mathbf{D}_{12}^{nl}$	2.8389	1.8123	1.7684	1.0271	0.3417
$\mathbf{D}_{13}^{nl}$	1.9872	0.9468	0.9990	1.0723	0.8056
$\mathbf{D}_{14}^{nl}$	1.1468	0.6732	0.9233	0.6187	0.7711
$\mathbf{D}_{15}^{nl}$	1.0127	0.7928	0.7748	0.7365	0.9882
Mean	2.1549	1.4760	1.4467	1.3489	1.1942

We execute 15 times data generations, and thus we obtain $\{\mathbf{D}_i^l, \mathbf{D}_i^{nl}, i = 1, \dots, 15\}$. We test our DSFT_l and DSFT_{nl} on all these datasets. Specifically, for DSFT_{nl}, we use different kernels including ‘polynomial’ kernel (poly), ‘inverse-cosine’ kernel (acos) and ‘radial basis function’ kernel (rbf). We compare to a baseline, COM, that only uses common features of the source and target domains in transfer to demonstrate the benefits of exploiting the domain specific features to transfer. We use “liblinear” for linear regression [34] as the base model, and we use mean squared error as the evaluation metric. The results on datasets $\{\mathbf{D}_i^l, i = 1, \dots, 15\}$ and $\{\mathbf{D}_i^{nl}, i = 1, \dots, 15\}$ are shown in Table 1 and Table 2, respectively.

From Table 1 and Table 2, it can be observed that all DSFT methods including DSFT_l and DSFT_{nl} with different nonlinear kernels achieve promising improvements on the adaptation performance compared to COM on both $\{\mathbf{D}_i^l, i = 1, \dots, 15\}$ and $\{\mathbf{D}_i^{nl}, i = 1, \dots, 15\}$. This demonstrates the effectiveness of the proposed general framework DSFT on the usage of the domain specific features for various adaptation tasks.

Compared the results of DSFT_l to those of DSFT_{nl}, we notice that DSFT_{nl} generally outperforms DSFT_l on both $\{\mathbf{D}_i^l, i = 1, \dots, 15\}$ and $\{\mathbf{D}_i^{nl}, i = 1, \dots, 15\}$. Specifically, regarding the

average result on datasets $\{\mathbf{D}_i^l, i = 1, \dots, 15\}$, DSFT_l beats DSFT_{poly} , but is clearly worse than DSFT_{acos} and DSFT_{rbf} . Considering that the relationship between common features and domain specific features is linear on datasets $\{\mathbf{D}_i^l, i = 1, \dots, 15\}$, we conclude that the nonlinearity exploration significantly boosts the effect of MMD minimization in DSFT. For datasets $\{\mathbf{D}_i^{nl}, i = 1, \dots, 15\}$ in which the relationship between common features and domain specific features is nonlinear, DSFT_l is whipped by DSFT_{nl} . It is reasonable as the nonlinearity exploration not only benefits the minimization of MMD, but also contributes to capturing the nonlinear relationship between features in this case.

Table 1 and Table 2 also show us the comparison results on DSFT_{nl} with different nonlinear kernels. Regarding the best adaptation performance on each dataset, the winner varies among DSFT_{poly} , DSFT_{acos} and DSFT_{rbf} . However, DSFT_{rbf} consistently achieves the best average result on both $\{\mathbf{D}_i^l, i = 1, \dots, 15\}$ and $\{\mathbf{D}_i^{nl}, i = 1, \dots, 15\}$. This is not unexpected as ‘rbf’ kernel has been shown very effective in both conventional machine learning and domain adaptation, especially when no prior knowledge is available. Thus, in our DSFT, we suggest a priority of using ‘rbf’ kernel. In the following experiments on real-world datasets, we utilize ‘rbf’ kernel as the default kernel for DSFT_{nl} .

5.2 Real-world Datasets & Experimental Setting

Amazon product review dataset [5] contains sentiment reviews from four product categories: books (B), DVD (D), electronics (E) and kitchen appliance (K). The task is a binary classification problem which predicts the sentiment of a review. Each review is labeled as positive or negative, and represented by the unigram and bigram tf-idf features. Each category has about 5,000 samples. About half of the samples are labeled as negative, and the rest are positive. A category is regarded as a domain. By pairing up any two domains, we have 6 domain pairs, denoted as ‘domain1-domain2’. By assigning one domain as source and the other as target, we construct one domain adaptation task, denoted as ‘source→target’. For example, in the domain pair B-D, B→D represents the domain adaptation task with B as source and D as target. Similarly we also have D→B with the source and the target flipped over.

20-Newsgroups [35] consists of about 20,000 documents characterized by word count features. They come from four top categories: computer (C), recording (R), science (S), and talk (T). Each top category has four subcategories. Top categories are treated as labels and subcategories are used to form related domains. The task is a binary classification problem which predicts the top category. We take computer (C) and science (S) as an example. Top category C is labeled as positive and top category S as negative. Two subcategories under each class are selected to form domain1, while the other two subcategories constitute domain2. By pairing up domain1 and domain2, we have one domain pair, denoted as ‘CS₁₋₂’. By exchanging the roles of the two domains, we have two domain adaptation tasks, denoted as ‘CS_{1→2}’ and ‘CS_{2→1}’.

Given a domain pair, common features are the features shared by the domain pair while domain specific features are those only appearing in a single domain. We use the top 1,000 most frequent common features in the domain pair and the top 1,000 most frequent domain specific features in each domain. Thus, the input feature space of one domain consists of 1,000 common features and 1,000 domain specific features. We take CS₁₋₂ as an example

to have a clear view of the common features and the domain specific features. The subcategory related to science is medicine in domain1, and electronics in domain2. The two subcategories share some common features, e.g., ‘science’, and ‘formula’. Each subcategory also has its own specific features, e.g., ‘molecular’ in medicine and ‘circuit’ in electronics.

Depending on the different strategies used to handle the hybrid feature space, we have two different baseline settings: ‘cut-off’ and ‘fill-up’. In ‘cut-off’, we use only the common features, denoted as COM. In ‘fill-up’, we fill in zeros to those specific features appearing in one domain but not the other, denoted as F_Zero. By doing ‘cut-off’ or ‘fill-up’, the feature spaces of the two domains become homogeneous, and we can apply existing homogeneous DA methods (HomoDA). We also compare our DSFT with such class of baselines. Specifically, in our experiments, we use state-of-the-art HomoDA methods: SA [23], GFK [7], FEDA [22], TKL [24], and TaR [25]. SVM [34] is used as the base classifier, and the prediction accuracy is used as the evaluation metric. We do cross validation on the source data to select the best hyperparameter C in $[10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3]$. For SA and GFK, we use the default subspace dimension d if it is specified by the authors, otherwise we set d to be the smallest one that contributes 85% of the energy. For TKL, we follow the authors’ hyperparameter configuration for text data, and for TaR, we use $p = 0.9$ and single layer with *tanh* activation function as stated by the authors. For DSFT_{nl} , we use rbf kernel.

5.3 DSFT v.s. ‘cut-off’

We compare DSFT_l and DSFT_{nl} with the ‘cut-off’ baseline and its variants ‘cut-off+HomoDA’ on the two benchmark datasets. The results are shown in Table 3. The best result on each adaptation task is highlighted in bold, and the runner-up is highlighted in bold-italic. We also show the average results of all the tasks in each dataset.

Table 3 shows that both DSFT_l and DSFT_{nl} consistently outperform COM on the two datasets, while the performance of COM variants varies in different tasks. DSFT_{nl} is a clear winner and DSFT_l achieves the runner-up in terms of the average performance among all the baselines. They achieve the best and the second best performance on most of the adaptation tasks. Specifically, DSFT_{nl} performs the best in 19 out of 24 tasks, and the second best in the 4 tasks. DSFT_l makes the best only in 2 tasks, but yields the second best in 18 tasks. These demonstrate the superiority of our DSFT to the ‘cut-off’ based baselines on the hybrid DA problem. Moreover, in general DSFT_{nl} outperforms DSFT_l . This is reasonable since the relationship between the common features and the domain specific features is more likely to be nonlinear in the complex real-world data. However, considering that DSFT_l already yields better average results than other baselines and even achieves the best in the ST₁₋₂ domain pair, it is also an effective and valuable method for hybrid DA problem.

Table 3 also shows that the improvement of DSFT over COM is more pronounced on the 20-Newsgroups dataset (DSFT_l : 4.49%; DSFT_{nl} : 6.07%) than that on the Amazon dataset (DSFT_l : 1.38%; DSFT_{nl} : 2.37%). We also observe marginal improvements (relative to the average) on the tasks E→K and K→E and CT_{1→2}.

Such an improvement variation among different domain pairs appears to be dataset/task-dependent. It would be interesting to investigate the reason behind. It is suspected that the variation may be an indication of different discriminative power of domain

Table 3
Cut-off comparison results (%)

Data	COM	GFK _C	SA _C	FEDA _C	TKL _C	TaR _C	DSFT _l	DSFT _{nl}
B → D	74.40	70.34	73.41	74.83	51.60	75.24	75.28	75.95
B → E	69.55	69.72	64.26	69.90	62.09	70.51	70.72	73.87
B → K	70.05	71.99	69.00	70.18	59.27	70.71	72.38	73.17
D → B	74.16	67.20	72.75	73.91	55.00	74.56	75.18	76.18
D → E	70.04	73.95	68.36	70.24	63.72	71.04	73.34	74.51
D → K	73.72	72.86	71.43	73.50	63.46	74.36	75.85	76.81
E → B	68.96	67.25	57.15	69.15	52.35	70.38	70.58	72.11
E → D	71.86	70.18	69.79	72.00	57.05	72.57	73.21	73.86
E → K	81.49	76.50	79.82	81.25	58.76	79.88	81.66	81.92
K → B	69.84	68.45	68.60	69.93	64.64	71.05	71.11	71.84
K → D	72.80	68.74	69.77	72.55	58.75	72.70	73.45	74.42
K → E	80.46	76.50	78.83	80.41	66.21	79.32	81.10	81.16
Mean	73.11	71.14	70.26	73.16	59.41	73.53	74.49	75.48
CR _{1→2}	76.50	77.27	72.04	79.59	64.49	72.64	83.28	87.48
CR _{2→1}	72.99	76.32	74.27	75.13	83.61	61.37	79.74	81.11
CS _{1→2}	80.17	82.15	79.06	81.55	80.96	72.53	85.24	88.50
CS _{2→1}	81.09	80.75	82.61	81.85	71.20	77.61	84.39	88.89
CT _{1→2}	88.55	82.21	86.68	90.32	57.01	86.47	90.11	91.16
CT _{2→1}	86.45	90.75	82.76	89.05	52.15	91.02	91.29	93.00
RS _{1→2}	84.21	93.62	84.80	85.22	83.21	89.42	88.33	91.77
RS _{2→1}	76.75	83.91	79.87	76.83	59.73	75.48	81.80	81.89
RT _{1→2}	81.21	75.33	74.01	82.70	56.07	84.16	84.64	85.60
RT _{2→1}	76.89	79.45	74.62	76.33	50.99	80.78	81.06	81.44
ST _{1→2}	71.35	69.92	71.73	70.40	56.06	64.23	74.48	74.10
ST _{2→1}	76.41	63.96	79.33	78.45	51.86	81.98	82.07	80.48
Mean	79.38	79.64	78.48	80.62	63.94	78.14	83.87	85.45

specific features in different tasks. We thus perform a quantitative analysis on it. To measure the discriminative power of domain specific features, we define a metric, P_{spec} , as follows. Given the data in a domain, we calculate the prediction accuracy under two cases: without domain specific features (i.e., using common features only); and with domain specific features (i.e., using all features). We conduct 5-fold cross-validation on the data to obtain the two accuracies, denoted as Acc_{wo} and Acc_w , respectively. The P_{spec} is then defined as:

$$P_{spec} = \frac{Acc_w - Acc_{wo}}{1 - Acc_{wo}}$$

A positive value of P_{spec} indicates that domain specific features carry the discriminative power and contribute to the prediction task. The larger the value is, the more powerful the discriminative capability is carried by domain specific features. Therefore, a larger P_{spec} gives more room for the performance improvement to be achieved when domain specific features are properly exploited.

Table 4 shows the Acc_{wo} , Acc_w and P_{spec} of each domain on the two datasets. In the Amazon dataset, BD_B denotes the domain ‘books’ in the domain pair B-D. In the 20-Newsgroups dataset, CR_1 denotes domain1 in the domain pair CR₁₋₂. From Table 4, we can see that P_{spec} of all domains are positive, which verifies the positive role that domain specific features play in the prediction. Also, P_{spec} values on 20-Newsgroups are much higher than those on Amazon, which explains why DSFT achieves more significant improvement on 20-Newsgroups. Interestingly, for those tasks on which DSFT achieves marginal improvements (E→K, K→E, and CT_{1→2}), the P_{spec} values of their target domains (EK_K, EK_E, and CT_2) are relatively small, compared to those of other domains in the same dataset. They indicate a

Table 4
Discriminative power analysis

Dom	Acc_{wo}	Acc_w	P_{spec}	Dom	Acc_{wo}	Acc_w	P_{spec}
BD_B	75.87	78.71	0.117	CR_1	86.57	93.08	0.477
BD_D	76.32	79.14	0.119	CR_2	95.02	97.00	0.397
BE_B	76.95	80.93	0.173	CS_1	90.47	93.19	0.286
BE_E	79.07	83.27	0.201	CS_2	89.53	93.48	0.377
BK_B	77.60	81.64	0.180	CT_1	92.88	95.41	0.354
BK_K	81.55	84.30	0.149	CT_2	93.33	94.79	0.219
DE_D	77.89	81.19	0.149	RS_1	89.87	95.36	0.542
DE_E	80.27	83.46	0.162	RS_2	95.21	97.73	0.526
DK_D	77.22	81.15	0.173	RT_1	88.15	93.18	0.424
DK_K	81.77	83.91	0.117	RT_2	89.87	95.33	0.539
EK_E	81.58	81.64	0.004	ST_1	89.82	94.16	0.426
EK_K	81.89	82.64	0.042	ST_2	88.19	93.14	0.419

small improvement room, which well explains the improvement variation of DSFT that we observed.

5.4 DSFT v.s. ‘fill-up’

Though domain specific features are shown to be useful, they should be exploited in a wise way in order to help with the adaptation. In the current practice, domain specific features are simply concatenated with common features, with features that do not appear in one domain filled with zeros. This corresponds to our baseline setting: F_Zero. We compare DSFT with this ‘fill-up’ baseline and its variants ‘fill-up+HomoDA’ on the two datasets. The results are reported in Table 5.

Table 5 shows that both DSFT_l and DSFT_{nl} consistently achieve better results than F_Zero in all adaptation tasks. For other variants, GFK₀, SA₀, FEDA₀, TKL₀, and TaR₀, the performance varies in different tasks. In some tasks, they can achieve significantly positive transfer, e.g., GFK₀ in ‘CS_{1→2}’ and ‘CS_{2→1}’, but they may also lead to negative transfer in other tasks, e.g., GFK₀ in ‘CT_{1→2}’ and ‘CT_{2→1}’. This is not unexpected as these methods are not specially designed for hybrid DA, and thus neglect to explicitly utilize the domain specific features for transfer. As F_Zero fills zeros to those domain specific features appearing only in one domain, the discriminative power embedded in these domain specific features may not be well exploited. Again, DSFT_{nl} achieves the best average performance and DSFT_l achieves the second best, which shows the superiority of DSFT to the ‘fill-up’ based baselines on the hybrid DA problem.

We also observe that the results of F_Zero are comparable to those of COM. On average, the accuracy of COM is 73.11% and that of F_Zero is 72.84% on Amazon dataset, and they are 79.38% and 79.42% on 20-Newsgroups dataset. This indicates that the use of domain specific features in F_Zero brings no benefit to the adaptation task. The reason is as follows. For F_Zero, the weights of SVM trained on source have three components, ω_{S_c} , ω_{S_s} and ω_{S_a} , each corresponding to one type of features in the extended feature space. Since S_a has all zeros in F_Zero, ω_{S_a} trained from S_a simply induces noise when tested on the target data. As for ω_{S_s} , it is trained from S_s , but the corresponding target feature matrix T_a is filled with zeros, which makes ω_{S_s} useless when doing the prediction. As a result, only ω_{S_c} is in effect and consequently domain specific features in F_Zero do not

Table 5
Fill-up comparison results (%)

Dataset	F_Zero	GFK ₀	SA ₀	FEDA ₀	TKL ₀	TrR ₀	DSFT _l	DSFT _{nl}
B → D	74.54	60.79	69.95	74.54	50.41	75.85	75.28	75.95
B → E	68.11	72.11	62.13	68.14	50.06	69.24	70.72	73.87
B → K	69.46	70.84	60.76	69.52	52.82	67.83	72.38	73.17
D → B	74.45	65.78	70.27	74.38	60.04	75.85	75.18	76.18
D → E	68.82	71.68	68.12	68.97	51.72	69.72	73.34	74.51
D → K	72.98	74.09	72.40	72.71	53.19	66.74	75.85	76.81
E → B	70.13	71.55	58.78	70.13	60.84	69.60	70.58	72.11
E → D	72.04	65.98	67.33	72.41	59.42	72.98	73.21	73.86
E → K	81.24	73.80	79.02	81.18	54.16	81.38	81.66	81.92
K → B	70.29	68.82	65.25	70.33	69.53	71.08	71.11	71.84
K → D	72.04	66.35	67.10	71.77	57.01	71.96	73.45	74.42
K → E	80.00	71.39	79.88	79.87	50.16	80.49	81.10	81.16
Mean	72.84	69.43	68.42	72.83	55.74	72.73	74.49	75.48
CR _{1→2}	80.36	81.88	73.33	78.22	66.64	59.43	83.28	87.48
CR _{2→1}	62.14	77.78	77.86	62.82	81.80	52.56	79.74	81.11
CS _{1→2}	77.68	86.78	84.21	78.80	84.81	51.07	85.24	88.50
CS _{2→1}	81.42	87.70	79.13	81.51	68.21	50.38	84.39	88.89
CT _{1→2}	89.91	88.76	82.83	90.74	89.59	70.03	90.11	91.16
CT _{2→1}	89.86	89.05	85.46	89.41	52.15	82.76	91.29	93.00
RS _{1→2}	85.22	86.99	80.69	85.14	77.01	70.36	88.33	91.77
RS _{2→1}	78.85	80.39	81.13	77.25	68.83	80.71	81.80	81.89
RT _{1→2}	80.95	80.95	80.60	82.44	56.07	83.23	84.64	85.60
RT _{2→1}	76.89	80.21	79.36	73.20	51.08	82.29	81.06	81.44
ST _{1→2}	71.35	71.73	75.90	70.87	56.06	74.38	74.48	74.10
ST _{2→1}	78.36	80.74	73.85	79.15	51.86	65.28	82.07	80.48
Mean	79.42	82.75	79.53	79.13	67.01	68.30	83.87	85.45

actually play a role. This explains why F_Zero and COM have similar performance.

5.5 Adaptation Analysis of DSFT

In this section, we investigate why DSFT is effective in exploiting the domain specific features in the adaptation from two aspects: (1) reweighting and (2) new feature representation. We use the 20-NewsGroup dataset in the empirical evaluation.

5.5.1 Feature Reweighting

DSFT reweights the common features by exploiting the domain specific features. By doing so, those common features that can bring the two domains closer are popped up and ranked higher. We verify this using DSFT_l as the shown case.

First, we show how DSFT_l reweights the common features. The final features of DSFT_l shown in Eq. (8). We approximate $\mathbf{S}_s = \mathbf{W}_s^T \mathbf{S}_c$, and $\mathbf{T}_t = \mathbf{W}_t^T \mathbf{T}_c$ (not exactly equal due to the discrepancy term), and we obtain:

$$\begin{aligned}\mathbf{X}_{s,h}^l &= [\mathbf{S}_c; \mathbf{W}_s^T \mathbf{S}_c; \mathbf{W}_t^T \mathbf{S}_c] = \mathbf{P} \mathbf{S}_c, \\ \mathbf{X}_{t,h}^l &= [\mathbf{T}_c; \mathbf{W}_s^T \mathbf{T}_c; \mathbf{W}_t^T \mathbf{T}_c] = \mathbf{P} \mathbf{T}_c, \\ \mathbf{P} &= [\mathbf{I}; \mathbf{W}_s^T; \mathbf{W}_t^T].\end{aligned}$$

We use the SVM weight as the feature importance for the two domains. For COM, we have:

$$Src : \mathbf{S}_c^T * \omega_{com_s}, \quad Tar : \mathbf{T}_c^T * \omega_{com_t}. \quad (18)$$

For DSFT_l, we have:

$$\begin{aligned}Src : \mathbf{X}_s^{h,T} \omega_{DSFT_{ls}} &= \mathbf{S}_c^T * \mathbf{P}^T \omega_{DSFT_{ls}}, \\ Tar : \mathbf{X}_t^{h,T} \omega_{DSFT_{lt}} &= \mathbf{T}_c^T * \mathbf{P}^T \omega_{DSFT_{lt}}.\end{aligned} \quad (19)$$

Table 6
Source training error (e-2) and A-distance

Metric	Source Training Error				A-distance			
Domain	COM	F_Zero	DSFT _l	DSFT _{nl}	COM	F_Zero	DSFT _l	DSFT _{nl}
CR _{1→2}	6.752	2.735	2.735	2.735	1.418	1.703	1.317	1.310
CR _{2→1}	3.173	1.286	1.115	1.286	1.417	1.683	1.314	1.312
CS _{1→2}	4.835	3.732	3.902	3.562	1.223	1.546	1.180	1.170
CS _{2→1}	5.571	3.262	3.262	3.262	1.203	1.514	1.154	1.135
CT _{1→2}	2.783	1.885	1.885	1.795	1.306	1.490	1.190	1.210
CT _{2→1}	2.706	2.289	2.393	2.185	1.288	1.476	1.197	1.190
RS _{1→2}	4.465	1.938	2.106	2.106	1.607	1.848	1.546	1.532
RS _{2→1}	2.435	0.756	0.924	0.840	1.600	1.849	1.566	1.552
RT _{1→2}	5.114	2.557	2.462	2.367	1.505	1.752	1.432	1.453
RT _{2→1}	3.600	1.844	1.756	1.668	1.500	1.762	1.420	1.444
ST _{1→2}	4.064	2.120	2.120	2.120	1.425	1.693	1.361	1.360
ST _{2→1}	5.028	2.182	1.992	2.087	1.441	1.705	1.339	1.328
Average	4.225	2.215	2.221	2.168	1.411	1.668	1.335	1.333

From Eq. (18) and Eq. (19), we can see that DSFT_l reweights the common feature importance ω_{com_s} and ω_{com_t} to $\mathbf{P}^T \omega_{DSFT_{ls}}$ and $\mathbf{P}^T \omega_{DSFT_{lt}}$ respectively.

Next, we verify that the two domains become closer after reweighting. We use the normalized discounted cumulative gain (NDCG) [36] to measure the similarity of two ranks. In our case, we have:

$$\gamma = \frac{DCG_s}{DCG_t}.$$

The closer to 1 γ is, the closer the feature importance of the source and target is. We calculate γ_{COM} by using ω_{com_s} and ω_{com_t} . Similarly, we calculate $\gamma_{Reweight}$ by using $\mathbf{P}^T \omega_{DSFT_{ls}}$ and $\mathbf{P}^T \omega_{DSFT_{lt}}$. The results are shown in Figure 3. From Figure 3, we can see that $\gamma_{Reweight}$ is consistently larger than γ_{COM} , which indicates the two domains indeed become closer after reweighting.

DSFT is more than what we state above since the discrepancy term is taken into account. By using the weights of the final homogeneous features directly (e.g., $\omega_{DSFT_{ls}}$ and $\omega_{DSFT_{lt}}$ for DSFT_l), we can calculate γ_{DSFT_l} and $\gamma_{DSFT_{nl}}$ for DSFT_l and DSFT_{nl}, respectively. As shown in Figure 3, we observe that both γ_{DSFT_l} and $\gamma_{DSFT_{nl}}$ consistently outperform $\gamma_{Reweight}$. This is because the discrepancy term brings the two domains further closer. Furthermore, $\gamma_{DSFT_{nl}}$ is larger than γ_{DSFT_l} , which indicates the performance improvements of DSFT_{nl} to DSFT_l.

5.5.2 New Feature Representation

According to [37], a good feature representation for DA should satisfy (1) reducing the source training error; (2) reducing A-distance. In this section, we verify that DSFT (DSFT_l and DSFT_{nl}) provides a good new feature representation reducing the source training error and A-distance simultaneously.

Followed by [37], A-distance is approximated as:

$$A_{dis} = 2(1 - 2\epsilon)$$

where ϵ is the generalization error on the problem of discriminating between the source and target samples. In our experiment, we separate the source and target data into training set and test set. We train a linear SVM on the training set and apply it to the test set. The recognition error is taken as ϵ . Table 6 shows the source training error and A-distance for COM, F_Zero, DSFT_l, and DSFT_{nl}.

Figure 3. NDCG for COM and DSFT

As shown in Table 6, both $DSFT_l$ and $DSFT_{nl}$ reduce the source training error and A-distance when compared to COM. We also observe that (1) $DSFT_l$ is comparable with F_Zero on the source training error but much better on A-distance, and (2) $DSFT_{nl}$ achieves smaller results on both the source training error and A-distance than F_Zero. These show that DSFT ($DSFT_l$ and $DSFT_{nl}$) provides a better representation for DA than both COM and F_Zero, and thus is superior in the DA task. Moreover, when compared $DSFT_{nl}$ to $DSFT_l$, $DSFT_{nl}$ yields smaller average results than $DSFT_l$ on both metrics, which further indicates the superior adaptation performance of $DSFT_{nl}$ to that of $DSFT_l$. When compared F_Zero to COM, we can see that the source training error is reduced but A-distance is increased. The joint effect to benefit the adaptation task is by chance, which implies that the adaptation performance of F_Zero is comparable with COM in general.

5.6 Property Analysis of DSFT

In this section, we study the proposed DSFT in three aspects: 1) sensitivity analysis on the balancing parameter α and β ; 2) the individual effect of the translation term and the discrepancy term; 3) the individual effect of the source specific features and the target specific features. All the analyses are done on both $DSFT_l$ and $DSFT_{nl}$.

5.6.1 Sensitivity Analysis

We conduct the sensitivity analysis with respect to α and β , where α takes values in the range $\{1, 10, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7\}$, and β takes values in $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10\}$. We take results of the task $RT_{1 \rightarrow 2}$ in $DSFT_l$ as a showcase in Figure 4. As shown in Figure 4, with the increase of α , the classification accuracy generally increases until it gets stable. The larger α is, the more sensitive the discrepancy term is to the objective function. When the empirical MMD gets stable, larger values of α cannot further improve the performance. For $RT_{1 \rightarrow 2}$, this happens when θ reaches 10^5 , which is also the case for all other domain adaptation tasks. For hyperparameter β , it is used to control the model complexity and avoid overfitting. With the change of β , we can see that the accuracy reaches the peak when β is between 0.5 and 1. In our experiments, we set $\alpha = 10^5$ and $\beta = 1$ as default for both $DSFT_l$ and $DSFT_{nl}$.

5.6.2 Term Analysis

We analyze the individual effect of the translation term and the discrepancy term, as well as their joint effect, with respect to the adaptation performance. For each of $DSFT_l$ and $DSFT_{nl}$,

Figure 4. Sensitivity analysis on $RT_{1 \rightarrow 2}$ in $DSFT_l$

Table 7
Average results (%) for term analysis

Method	F_Zero	$DSFT_{l_T}$	$DSFT_{l_D}$	$DSFT_l$
Accuracy	76.26	78.29	77.39	79.18
Method	F_Zero	$DSFT_{nl_T}$	$DSFT_{nl_D}$	$DSFT_{nl}$
Accuracy	76.26	79.91	78.24	80.47

We develop two variants: one with the translation term but not the discrepancy term ($DSFT_{l_T}$ and $DSFT_{nl_T}$), the other with the discrepancy term but not the translation term ($DSFT_{l_D}$ and $DSFT_{nl_D}$). We compare the proposed $DSFT_l$ and $DSFT_{nl}$ with F_Zero and the corresponding two variants. We show the average accuracy of all the tasks on the two datasets in Table 7.

Table 7 shows that the two variants in both $DSFT_l$ and $DSFT_{nl}$ improve the accuracy of F_Zero, which demonstrates the effectiveness of each individual term in DSFT. Further more, integrating these two terms together, which is $DSFT_l$ or $DSFT_{nl}$, is able to further boost the improvement.

5.6.3 Domain Specific Feature Analysis

The final feature representation consists of the common features, the source specific features, and the target specific features. In this section, we discuss how the individual domain specific features affect the adaptation performance.

For each of $DSFT_l$ and $DSFT_{nl}$, we design two more variants: one is only combining the common features with the source

Table 8
Domain specific feature analysis

Method	COM	S_DSFT _l	T_DSFT _l	DSFT _l
Accuracy	76.25	77.15	78.04	79.18
Method	COM	S_DSFT _{nl}	T_DSFT _{nl}	DSFT _{nl}
Accuracy	76.25	77.66	79.72	80.47

specific features (S_DSFT_l and S_DSFT_{nl}), and another one is only combining the common features with the target specific features (T_DSFT_l and T_DSFT_{nl}). We compare DSFT_l and DSFT_{nl} with the corresponding two variants in terms of the adaptation performance on the two datasets. We also take the results of COM as a reference. We show the average accuracy of all the tasks in Table 8.

Table 8 shows that DSFT is the best, followed by T_DSFT, S_DSFT, and lastly COM in both linear and nonlinear cases. This illustrates that combining common features with specific features from any domain can improve the performance of COM. Concatenation of all the features together can achieve the best results. It also shows that the target specific features can bring more benefit to the adaptation performance than the source specific ones. This is because the prediction task is on the target domain, and thus the target features are more informative.

6 CONCLUSION

In this paper, we investigate a new problem setting, hybrid DA, and design a general framework DSFT to transfer knowledge across domains without any supplementary information. It learns the translations between common features and domain specific features, and utilizes the learned translations to construct a homogeneous feature space, in which cross domain knowledge is transferred and the domain divergence is minimized. We instantiate the DSFT framework to a linear case and a nonlinear case, i.e. DSFT_l and DSFT_{nl}. Extensive experiments show that both DSFT_l and DSFT_{nl} outperform COM and F_Zero as well as their couplings with state-of-the-art methods. The underlying mechanisms of DSFT, feature reweighting and new feature representation, are empirically investigated. It is also shown that DSFT benefits from both the translation term and the discrepancy term. The positive role that domain specific features play in DA is validated.

ACKNOWLEDGMENTS

This work was conducted within the Rolls-Royce@Nanyang Technological University Corporate Lab with support from the National Research Foundation (NRF) Singapore under the Corp Lab@University Scheme.

REFERENCES

- [1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [2] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 513–520.
- [3] S. J. Pan and Q. Yang, "A survey on transfer learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, 2010.
- [4] A. Margolis, "A literature review of domain adaptation with unlabeled data," in *Report Technique, University of Washington*, 2011, pp. 1–42.
- [5] J. Blitzer, M. Dredze, F. Pereira *et al.*, "Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification," in *ACL*, vol. 7, 2007, pp. 440–447.
- [6] R. Xia, J. Jiang, and H. He, "Distantly supervised lifelong learning for large-scale social media sentiment analysis," *IEEE Transactions on Affective Computing*, vol. 8, no. 4, pp. 480–491, 2017.
- [7] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2066–2073.
- [8] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1410–1417.
- [9] A. DInnocente, F. M. Carlucci, M. Colosi, and B. Caputo, "Bridging between computer and robot vision through data augmentation: a case study on object recognition," in *International Conference on Computer Vision Systems*. Springer, 2017, pp. 384–393.
- [10] F. Wu and Y. Huang, "Sentiment domain adaptation with multiple sources," in *Proceedings of the 54th Annual Meeting on Association for Computational Linguistics*, 2016, pp. 301–310.
- [11] P. Wei, Y. Ke, and C. K. Goh, "Deep nonlinear feature coding for unsupervised domain adaptation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 2189–2195.
- [12] Z. Yang, R. Salakhutdinov, and W. W. Cohen, "Transfer learning for sequence tagging with hierarchical recurrent networks," *International Conference on Learning Representation*, pp. 1–10, 2017.
- [13] C. Wang and S. Mahadevan, "Heterogeneous domain adaptation using manifold alignment," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, pp. 1541–1546.
- [14] W. Dai, Y. Chen, G.-R. Xue, Q. Yang, and Y. Yu, "Translated learning: Transfer learning across different feature spaces," in *Advances in neural information processing systems*, 2008, pp. 353–360.
- [15] J. T. Zhou, S. J. Pan, I. W. Tsang, and Y. Yan, "Hybrid heterogeneous transfer learning through deep learning," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 2213–2220.
- [16] 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.
- [17] J. Pan, X. Hu, P. Li, H. Li, W. He, Y. Zhang, and Y. Lin, "Domain adaptation via multi-layer transfer learning," *Neurocomputing*, vol. 190, pp. 10–24, 2016.
- [18] P. Wei, R. Sagarna, Y. Ke, Y.-S. Ong, and C.-K. Goh, "Source-target similarity modelings for multi-source transfer gaussian process regression," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, pp. 3722–3731.
- [19] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction," in *AAAI*, vol. 8, 2008, pp. 677–682.
- [20] B. Chen, W. Lam, I. Tsang, and T.-L. Wong, "Extracting discriminative concepts for domain adaptation in text mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 179–188.
- [21] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *Neural Networks, IEEE Transactions on*, vol. 22, no. 2, pp. 199–210, 2011.
- [22] H. Daume III, "Frustratingly easy domain adaptation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Association for Computational Linguistics, 2009, pp. 256–263.
- [23] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2960–2967.
- [24] M. Long, J. Wang, J. Sun, and P. S. Yu, "Domain invariant transfer kernel learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 27, no. 6, pp. 1519–1532, 2015.
- [25] S. Clinchant, G. Csurka, and B. Chidlovskii, "A domain adaptation regularization for denoising autoencoders," in *The 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 26–31.
- [26] X. Shi, Q. Liu, W. Fan, P. S. Yu, and R. Zhu, "Transfer learning on heterogeneous feature spaces via spectral transformation," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 1049–1054.

- [27] B. Kulis, K. Saenko, and T. Darrell, "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1785–1792.
- [28] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 6, pp. 1134–1148, 2014.
- [29] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. 49–57, 2006.
- [30] P. L. Q. W. Y. X. W. Z. Hongliang Yan, Yukang Ding, "Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation," in *International Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2272–2281.
- [31] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers, "Associative domain adaptation," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 6–14.
- [32] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International Conference on Computational Learning Theory*. Springer, 2001, pp. 416–426.
- [33] X. Niyogi, "Locality preserving projections," in *Neural information processing systems*, vol. 16. MIT, 2004, pp. 153–160.
- [34] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.
- [35] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 210–219.
- [36] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.
- [37] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in Neural Information Processing Systems 20*. Cambridge, MA: MIT Press, 2007, pp. 137–144.

Chi Keong Goh currently holds the position of R&T Team Lead, Computational Engineering, at the Rolls-Royce Applied Technology Group (ATG) in Singapore. Prior to joining the ATG, he was a senior research fellow in the Data Storage Institute, Agency for Science, Technology and Research (A*Star). His research interest includes ensemble and transfer learning, extraction and incorporation of expert knowledge into data analytic, and multi-objective optimization.

Pengfei Wei is currently working toward the PhD degree in the school of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include data analysis, transfer learning, and domain adaptation.

Yiping Ke is an assistant professor in the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore. Before joining NTU, she worked as a scientist with the Institute of High Performance Computing, A*STAR and a research assistant professor at Chinese University of Hong Kong. Her research interests include big data analysis, data mining, data management, and applications.