ADAPTIVE COMPONENT EMBEDDING FOR UNSUPERVISED DOMAIN ADAPTATION

Mengmeng Jing[1], *Jingjing Li*[1], *Ke Lu*[1], *Jieyan Liu*[1], *Zi Huang*[2]

[1]University of Electronic Science and Technology of China,
[2]The University of Queensland
jingmeng1992@gmail.com;lijin117@yeah.net;{liujy,kel}@uestc.edu.cn;huang@itee.uq.edu.au

ABSTRACT

Domain adaptation has obtained considerable interest from the literatures of multimedia, especially in cross-domain knowledge transfer problems. In this paper, we propose an effective yet time-saving approach, named Adaptive Component Embedding (ACE), for unsupervised domain adaptation. Specifically, ACE learns adaptive components across domains to embed all data in a shared subspace where the distribution divergence is mitigated and the underlying geometric structures in the local manifold are preserved. Then, an adaptive classifier is learned by using Representer Theorem in the Reproducing Kernel Hilbert Space (RKHS). The objective of our method can be efficiently solved in a closed form. Comprehensive experiments on both standard and large-scale datasets verify that ACE significantly outperforms previous state-of-the-art methods in terms of the classification accuracy and training time.

Index Terms— Domain adaptation, transfer learning, domain alignment, subspace learning

1. INTRODUCTION

In real-world multimedia applications, e.g., cross-modal multimedia retrieval, data used to train the model may significantly differ from the test data. Therefore, the traditional machine learning methods would fail to handle these cross-domain tasks as the source and target data are drawn from different distributions [1, 2, 3, 4]. In recent years, domain adaptation (DA) [5, 6, 7, 8, 9] shows obvious advantages in solving such a challenging problem.

Existing work [1, 3, 5, 10, 11] on DA generally learn a domain-invariant feature representation to align the source and target domains. The learned feature representation can either reduce the distribution divergences between domains [1, 2], or preserve some important properties [3, 12], e.g., geometric structures and statistical properties. However, these methods generally preserve key properties or align the distributions independently, which is suboptimal when the distribution divergence is substantially large [1, 3]. Thus, it is worth learning a more effective feature representation which can not only preserve key properties but also align distributions.

Recently, some researches have already witnessed remarkable performance via jointly optimizing the property preservation and distribution alignment [10, 5, 13]. However, these methods are very complex and time-consuming. In real-world multimedia applications, such as online cross-modal retrieval [14], time is vital for the success of the application. Therefore, an effective and, at the same time, efficient DA method is absolutely required and valuable. Unfortunately, the speed and accuracy are a pair of contradictions.

In this paper, we present a novel DA method, i.e., Adaptive Component Embedding (ACE), which seeks the adaptive components to reduce discrepancies between domains. The proposed method achieves an exquisite balance between the the speed and accuracy. Specifically, in terms of promoting performance, the adaptive components are firstly learned by ACE. Data in both domains are projected onto a latent subspace spanned by these adaptive components. In this subspace, the distribution divergence is minimized and the geometric structures in the ambient space are preserved. Then, a transformation matrix is computed to further align the covariance of both domains. Finally, an adaptive classifier is learned by minimizing the structural risk in the RKHS space. On the other hand, regarding the time complexity, we carefully formulate our method so that the most time-consuming operation is independent of the number of data. In addition, ACE has a closed-form solution and avoids tedious iterations. To summary, the main contributions of this paper are:

(1) We learn the adaptive components across domains to embed data in a domain-invariant feature subspace, where the distance of distribution is reduced and the geometric structures underlying the local manifold are preserved.

(2) We seek a balance between high performance and low time-consuming. Compared with the extremely simple methods, such as CORAL [3] and TCA [1], ACE consumes a bit more time to achieve a significant performance improvement. While compared with the complex methods, such as JGSA [5] and JDA [2], ACE saves a large amount of training time yet achieves better performance. ACE is applicable to large-scale datasets.

(3) Extensive experiments on three real-world DA benchmarks which include a large-scale dataset verify that our

978-1-5386-9552-4/19/$31.00 ©2019 IEEE
DOI 10.1109/ICME.2019.00286

Table 1: Algorithms comparison.

Properties	TCA [1]	JDA [2]	JGSA [5]	ARTL [10]	ACE (Ours)
Robust				√	√
Distribution aligned	√	√	√	√	√
Pseudo-label free	√				√
Locality consistency				√	√
Iterations free	√			√	√
Covariance aligned					√

method can outperform state-of-the-art methods with significant advantages in light of both accuracy and speed.

2. RELATED WORK

The most related methods with ACE are TCA [1], JDA [2], JGSA [5] and ARTL [10]. For a clear understanding, we list the main properties of them in Table 1. The main differences of these methods are highlighted as follows: 1) Only ARTL and ACE preserve the local neighborhood information of samples when project data onto the subspaces; 2) In addition to aligning the marginal distribution, JDA, JGSA and ARTL also align the conditional distribution by optimizing the conditional MMD [2]. Since computing the conditional MMD need target labels, these methods obtain pseudo labels through basic classifiers. The pseudo labels, however, are less reliable. JDA and JGSA employ multiple iterations to refine the labels, which obviously increases the runtime of the methods. 3) Both ARTL and ACE learn the classifiers by optimizing the structural risk minimization (SRM) [15]. Nevertheless, ARTL learns the transformation matrix and classifier in one step, while ACE learns the adaptive components to get the domain-invariant feature representation firstly, then trains the classifier through optimizing SRM; 4) ACE also matches the covariance of both domains, which is an important second-order statistics in DA.

Recently, deep neural networks [16] are introduced to promote the performance of DA [6, 7, 17]. DDC [6] is a DA variant of CNN that maximize the domain invariance by regularizing the adaptation layer of AlexNet using linear-kernel MMD. DAN [7] further extends DDC by embedding deep features to RKHS space and matching different distributions optimally using multi-kernel MMD.

3. THE PROPOSED METHOD

3.1. Problem Definition

Let $X_s \in \mathcal{R}^{m\times n_s}$ and $X_t \in \mathcal{R}^{m\times n_t}$ be the source and target domain, respectively, where m is the original dimensionality of the data samples, n_s and n_t are numbers of the source and target samples. Data in the source and target domain are drawn from probability distribution $\mathcal{P}(X_s)$ and $\mathcal{P}(X_t)$, respectively. In this paper, we aim to tackle the unsupervised DA, where the source domain label Y_s is provided, while the target domain label Y_t is unknown. ACE learns a classifier f in the source domain data, and utilizes it into the target domain data with the condition that $\mathcal{P}(X_s) \neq \mathcal{P}(X_t)$.

3.2. Problem Formulation

In DA, discrepancies between the source and target domains are significantly large. Therefore, transferring without appropriate alignment will cause poor performance. In this paper, our proposed method has two main steps, one is learning the new feature representation, the other is learning an adaptive classifier. Specifically, ACE first learns the adaptive components $P \in \mathcal{R}^{m\times d}$ (d < m) to embed the source and target domain data in a shared domain-invariant subspace and thus get the new feature representation. Then, ACE aligns the covariance of the new feature representation. Finally, the classifier f is learned by optimizing SRM. For a global understanding, the objective function is:

$$\begin{aligned}\arg\min_{f\in\mathcal{H}_\mathcal{K},P} &\sum_{i=1}\ell(f(P^Tx_i),y_i)+\eta||f||_K^2\\ &+G(X_s,X_t,P)+\Omega(P^TX_s,P^TX_t),\end{aligned}\tag{1}$$

where f is the robust adaptive classifier, ℓ is a loss function, G stands for regularization terms used to learn P, Ω aligns the covariance of the new feature representations, $\mathcal{H}_K$ represents a set of classifiers in the RKHS space, $||f||_K^2$ denotes the squared norm of f in $\mathcal{H}_K$, η is the trade-off parameter. In the remainder of this section, we will report each part in detail and show how to optimize the objective.

3.2.1. Distribution Alignment

To reduce the distribution discrepancies between domains, it is necessary to find a proper distance measure so that the distribution discrepancies can be quantified. In this paper, we adopt the empirical MMD [18] as the distance measure, which computes the difference between means of both domains:

$$\arg\min_P \left\|\frac{1}{n_s}\sum_{i=1}^{n_s}P^Tx_s^i-\frac{1}{n_t}\sum_{j=1}^{n_t}P^Tx_t^j\right\|_F^2.\tag{2}$$

The equivalent matrix form of (2) is:

$$\arg\min_P \mathrm{Tr}(P^TXM_0X^TP),\tag{3}$$

where M_0 is the marginal MMD matrix written as:

$$(M_0)_{ij}=\begin{cases}\frac{1}{n_sn_s}, & x_i,x_j\in X_s,\\ \frac{1}{n_tn_t}, & x_i,x_j\in X_t,\\ \frac{-1}{n_sn_t}, & \text{otherwise.}\end{cases}\tag{4}$$

3.2.2. Local Consistency

We deploy the local consistency regularization to preserve the geometric structures among samples. Specifically, we construct the similarity matrix W to characterize the sample relationships. If node i is among the node j's k-nearest neighbors, $W_{ij}=e^{-||x_i-x_j||^2/t}$; Otherwise $W_{ij}=0$. Then, the local consistency regularization term can be computed as:

$$\arg\min_P \mathrm{Tr}(P^TX\tilde{L}X^TP),\tag{5}$$

where $\tilde{L}=I-D^{-1/2}LD^{-1/2}$ is the normalized Laplacian matrix, $L=D-W$ is the Laplacian matrix, $D_{ii}=\sum_{j\neq i}W_{ij}$, $X=[X_s, X_t]$. Using the normalized Laplacian matrix $\tilde{L}$ instead of the unnormalized one L provides certain theoretical guarantees and seems to perform as well or better in many practical tasks [19].

3.2.3. Adaptive Component Embedding

To learn P, we incorporate (3), (5) and get the optimization function, which is equivalent with the $G(X_s, X_t, P)$ in (1). We follow [1] to add a constraint term $\mathrm{Tr}(P^T P)$ to further control the complexity of P. Then, $G(X_s, X_t, P)$ becomes:

$$\begin{aligned} \arg\min_{P} \mathrm{Tr}(P^T X M_0 X^T P) + \rho \mathrm{Tr}(P^T X \tilde{L} X^T P) + \lambda \mathrm{Tr}(P^T P), \\ \text{s.t. } P^T X H X^T P = I, \end{aligned} \tag{6}$$

where $\lambda, \rho > 0$ are two trade-off parameters to balance the contribution of different parts. Then, $H = I - \frac{1}{n}\mathbf{1}$ is the centering matrix, $\mathbf{1} \in \mathcal{R}^{n \times n}$ is the matrix with all 1s, $I \in \mathcal{R}^{n \times n}$ is the identity matrix and n is the total number of all data. We add $P^T X H X^T P = I$ to avoid the trivial solutions, e.g., P = 0. To learn new features spanned by P, we first get the Lagrange function of (6):

$$\begin{aligned} \mathcal{L} = & \ \mathrm{Tr}(P^T (X M_0 X^T + \rho X \tilde{L} X^T + \lambda I) P) \\ & + \mathrm{Tr}((P^T X H X^T P - I)\Phi), \end{aligned}$$

where $\Phi = \mathrm{diag}(\lambda_1, \cdots, \lambda_d)$ are the d largest eigenvalues.Then, by setting the derivative $\frac{\partial \mathcal{L}}{\partial P}$= 0, we get:

$$(X M_0 X^T + \rho X \tilde{L} X^T + \lambda I) P = X H X^T P \Phi, \tag{7}$$

where $P = [P_1, \cdots, P_d]$ contains the corresponding eigenvectors, which are just the adaptive components and can be obtained through solving eigen-decomposition.

At last, we get the domain-invariant feature representations by:

$$Z_s = P^T X_s \text{ and } Z_t = P^T X_t. \tag{8}$$

3.2.4. Covariance Alignment

As the source and target domains have different statistics, e.g., mean and variance, features are usually normalized to have zero mean and unit variance before training a classifier on them. However, Sun et al. [3] report that only normalizing the mean and variance are not good enough to reduce the distribution difference. Aligning the second-order statistics, covariance, is also essential. Denoting the covariance matrices of the source and target feature representation by C_s and C_t respectively, the covariance distance can be computed as:

$$\Omega = ||C_{\tilde{s}} - C_t||_F^2, \tag{9}$$

where $C_{\tilde{s}}$ is the transformed covariance matrix of the source feature representation Z_s. It is noteworthy that numbers of the source and target domain may be different, which causes size mismatch of two covariance matrices. To solve this problem, we learn an adaptation transformation Q to make the source domain closer to the target domain:

$$\Omega = ||Q^T C_s Q - C_t||_F^2. \tag{10}$$

Algorithm1: Adaptive Component Embedding

Input: source and target domain data: X_s, X_t; labels for source domain data:Y_s, subspace dimensionality d, regularization parameters ρ, η and λ.
Output: Classification result $\tilde{Y}_t$
begin
1: Optimize (7) and get the new representation $Z_s = P^T X_s$ and $Z_t = P^T X_t$;
2: Compute the alignment matrix Q through (11), and further get $\tilde{Z}_s = Z_s Q$;
3: Construct the kernel matrix K by using $\tilde{Z}_s$, Z_t;
4: Compute the coefficient matrix β by solving (20);
5: Get classification result $\tilde{Y}_t$ through (15);
end

Following [3], Q is empirically computed as:

$$Q = (C_s + I_{n_s})^{-\frac{1}{2}} (C_t + I_{n_t})^{\frac{1}{2}}, \tag{11}$$

where $I_{n_s} \in R^{n_s \times n_s}$ and $I_{n_t} \in R^{n_t \times n_t}$ are the identity matrices. Then, we can align Z_s to be close with Z_t by:

$$\tilde{Z}_s = Z_s Q. \tag{12}$$

3.2.5. Learning the Adaptive Classifier

We use the least squares as the loss function ℓ in (1). Thus, we have:

$$\arg\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n_s} (y_i - f(z_i))^2 + \eta ||f||_K^2. \tag{13}$$

Using the classical Representer Theorem [20], the solution of (13) exists in RKHS and can be written as:

$$f^*(z) = \sum_{i=1}^{n_s} \beta_i K(z_i, z), \tag{14}$$

where K is a kernel function in RKHS. To train a more effective classifier, the target data should be exploited as well. Thus, we use the revised Representer Theorem [21] to expand f as follows:

$$f^*(z) = \sum_{i=1}^{n} \beta_i K(z_i, z), \tag{15}$$

where $\beta = (\beta_1, \beta_2, ..., \beta_n)^T \in \mathcal{R}^{n \times c}$ is the classifier parameters, c is the number of all classes. Then, (13) can be rewritten as:

$$\arg\min_{f \in \mathcal{H}_K} ||(Y - \beta^T \tilde{K})\Lambda||_F^2 + \eta ||f||_K^2, \tag{16}$$

where $Y = [\tilde{Y}_s, 0_{c \times n_t}]$ is an indicator matrix where $\tilde{Y}_s \in \mathcal{R}^{c \times n_s}$ are the soft labels for source domain. If x_s^i belongs to class j, $\tilde{Y}_s(i,j) = 1$; Otherwise $\tilde{Y}_s(i,j) = 0$. $\tilde{K} \in \mathcal{R}^{n \times n}$ is the kernel matrix with $\tilde{K}(i,j) = K(z_i, z_j)$. In (16), Λ is a diagonal matrix with $\Lambda_{ii} = 1$ if $i \leqslant n_s$ else $\Lambda_{ii} = 0$.

Table 2: Accuracy (%) on Office+Caltech with SURF features.

X_s	X_t	1NN	PCA	SVM	TCA [1]	CORAL [3]	JDA [2]	JGSA [5]	ARTL [10]	ACE
C	D	25.5	44.6	47.8	45.9	40.7	49.0	45.9	39.5	**56.1**
	W	25.8	34.6	41.7	39.3	39.2	39.3	45.4	31.5	**50.5**
	A	23.7	39.5	53.1	45.6	47.2	43.1	51.5	44.1	**56.8**
A	D	25.5	33.8	44.6	35.7	38.3	42.0	**47.1**	36.9	45.2
	W	29.8	35.9	31.9	40.0	38.7	38.0	45.8	33.6	**51.5**
	C	26.0	39.0	41.7	42.0	40.3	40.9	41.5	36.1	**44.7**
W	D	59.2	89.2	78.3	91.1	84.9	92.4	90.5	87.9	**93.0**
	A	23.0	29.1	27.6	30.5	37.8	29.8	39.9	38.3	**42.4**
	C	19.9	28.2	28.8	31.5	34.6	33.0	33.2	29.7	**34.5**
D	W	63.4	86.1	52.5	87.5	85.9	89.2	91.9	88.5	**92.5**
	A	28.5	33.2	26.2	32.8	38.1	33.4	38.0	34.9	**41.3**
	C	26.3	29.7	26.4	33.0	34.2	31.2	29.9	30.5	**34.3**
Avg.		31.4	43.6	41.1	46.2	46.7	46.8	50.0	44.3	**53.6**

Considering classifier function of the form $f(\cdot) = \sum_{i=1}^{n} \beta_i K(z_i, \cdot)$, we can get:

$$||f||_K^2 = \langle f, f \rangle_K = \left\langle \sum_{i=1}^{n} \beta_i K(z_i, \cdot), \sum_{j=1}^{n} \beta_j K(z_j, \cdot) \right\rangle_K = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i^T \beta_j \langle K(z_i, \cdot), K(z_j, \cdot) \rangle_K. \quad (17)$$

Using the following reproducing kernel property of RKHS: $\langle K(x, \cdot), K(y, \cdot) \rangle_K = K(x, y)$, (17) can be further derived as:

$$||f||_K^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_i^T \beta_j K(z_i, z_j) = \mathrm{Tr}(\beta^T \tilde{K} \beta). \quad (18)$$

Therefore, (16) can be rewritten as:

$$\arg\min_{f \in \mathcal{H}_K} ||(Y - \beta^T \tilde{K})\Lambda||_F^2 + \eta \mathrm{Tr}(\beta^T \tilde{K} \beta). \quad (19)$$

For β, the above function is convex. Thus, by setting the derivative of (19) to 0, we get:

$$\beta = (\Lambda \tilde{K} + \eta I)^{-1} \Lambda Y^T. \quad (20)$$

3.2.6. Inference

After learning the domain-invariant kernel K and the classifier coefficients β, we can get the soft labels of the target domain data through (15). For clarity, we show the main steps of our method in Algorithm 1.

3.3. Time Complexity

The most time-consuming part in ACE is solving eigen-decomposition problems in step 1, whose time complexity is $O(dm^2)$. Obviously, this time-consuming operation is independent of the number of data. In addition, matrix multiplication and inverse is the very common operation in most of DA methods whose time complexity is $O(n^{2.3757})$ by the Coppersmith-Winograd method [22].

4. EXPERIMENTS

This section presents the experimental results of ACE. Following the previous work [1], all of the results in this paper are in terms of the accuracy on the target domain: $|x : x \in X_t \wedge \hat{y}_t = y_t|/|x : x, \in X_t|$, where x is a target sample, y_t is the real label for x, $\hat{y}_t$ is the predicted label for x. ACE is compared with 10 state-of-the-art DA methods. In addition, we also compare our method with 3 deep DA approaches [16, 7, 6]. For all the compared baselines, we either report the best results from their original paper or the results we can achieve by running their codes. For ACE, we fix $\lambda = 10$, $\eta = 0.5$. Then, d and ρ are the hyper-parameters and will be reported in the following subsection.

4.1. Datasets and Experimental Settings

Office + Caltech contains 4 domains and each domain has 10 categories of object images [23, 24]. These four domains are C (Caltech), A (Amazon), W (Webcam) and D (DSLR). The original images are processed and transformed to two types of features, e.g. 800-dimensional SURF and 4096-dimensional DeCAF$_6$ [25]. For the hyper-parameters, we set $d = 35$, $\rho = 5$ for evaluations on SURF features, and set $d = 20$, $\rho = 25$ for evaluations on DeCAF$_6$ features.

Office + Home [17] dataset includes 4 domains, i.e., Art, Clipart, Product and Real-World. There are about 15500 images in total. Each domain contains 65 classes. We use a VGG-F model pretrained using the ImageNet 2012 to retrieve the 4096-dimensional deep features, which are the fc7 layer output of the model. As for the hyper-parameters, we set $d = 200$, $\rho = 5$ for evaluations on Office+Home dataset.

4.2. Experimental Results and Analysis

Experimental results of ACE and the compared baselines on 3 benchmarks are reported in Table 2- 4, respectively.

On Office + Caltech with SURF features, ACE achieves the best performance on 11 out of 12 tasks compared with other baselines. The only one exception is on A $\rightarrow$ D, on which ACE still achieves the second best performance. The average classification accuracy of ACE is 53.6%, obtaining 3.6% improvement compared with the best baseline JGSA. It is noteworthy that the accuracy of the basic classifier 1NN is 31.4%, which is the worst result in all of the baselines. Therefore, some pseudo-label-based methods, i.e., JDA, JGSA and ARTL, which use the basic classifier to get the initial pseudo labels, may get performance degradation due to the extremely

Table 3: Accuracy (%) on Office + Caltech with DeCAF$_6$ features.

X_s	X_t	Traditional Methods					Deep Methods			ACE
		TCA [1]	CORAL [3]	JDA [2]	JGSA [5]	ARTL [10]	AlexNet [16]	DAN [7]	DDC [6]	
C	D	85.4	84.7	89.8	93.6	86.6	87.1	89.3	88.8	**94.3**
	W	78.3	80.0	85.1	86.8	87.8	83.7	90.6	85.4	**93.9**
	A	89.8	92.0	89.6	91.4	92.4	91.9	92.0	91.9	**93.3**
A	D	81.5	84.1	80.3	88.5	85.4	87.4	91.7	89.0	**93.0**
	W	74.2	74.6	78.3	81.0	88.5	79.5	**91.8**	86.1	89.5
	C	82.6	83.2	83.6	84.9	**87.4**	83.0	84.1	85.0	87.2
W	D	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
	A	84.1	81.2	90.3	90.7	**92.3**	83.8	92.1	84.9	91.9
	C	80.4	75.5	84.8	85.0	**88.2**	73.0	81.2	78.0	86.3
D	W	99.7	99.3	99.7	99.7	**100.0**	97.7	98.5	98.2	**100.0**
	A	89.1	85.5	91.7	92.0	**92.7**	87.1	90.0	89.5	**92.7**
	C	82.3	76.8	85.5	86.2	87.3	79.0	80.3	81.1	**87.7**
Avg.		85.6	84.7	88.2	90.0	90.7	86.1	90.1	88.2	**92.5**

Table 4: Accuracy (%) on Office + Home dataset.

Xs	Xt	TCA	CORAL	JDA	DAN	DANN[26]	ACE
Ar	Cl	19.9	27.1	25.3	30.7	**33.3**	32.5
	Pr	32.1	36.2	36.0	42.2	43.0	**49.2**
	Rw	35.7	44.3	42.9	54.1	54.4	**54.7**
Cl	Ar	19.0	26.1	24.5	32.8	32.3	**37.0**
	Pr	31.4	40.0	40.2	47.6	49.1	**51.0**
	Rw	31.7	40.3	40.9	49.8	49.8	**51.1**
Pr	Ar	21.9	27.8	26.0	29.1	30.5	**35.6**
	Cl	23.6	30.5	32.7	34.1	**38.1**	32.9
	Rw	42.1	50.6	49.3	56.7	56.8	**58.2**
Rw	Ar	30.7	38.5	35.1	43.6	44.7	**47.4**
	Cl	27.2	36.4	35.4	38.3	**42.7**	39.4
	Pr	48.7	57.1	55.4	62.7	64.7	**65.9**
Avg.		30.3	37.9	37.0	43.5	44.9	**46.2**

Table 5: Runtime (seconds) of different methods.

Datasets	CORAL	TCA	JDA	JGSA	ARTL	ACE
Office (SURF)	3.16	4.70	5.21	16.87	6.97	7.57
Office (DeCAF$_6$)	317.31	65.20	88.16	304.81	20.81	75.12
Office+Home	652.97	208.16	396.71	1095.23	854.86	369.80

low classification accuracy of the pseudo labels. Our ACE is designed to have no demand to rely on basic classifiers to get the initial pseudo labels. Hence, ACE avoids the influence of low performance of the basic classifiers and get the best classification accuracy.

On Office + Caltech with DeCAF$_6$ features, all of the methods gain a remarkable improvement compared with performance on SURF features. Obviously, deep neural networks can extract discriminative features more effectively than the traditional machine learning methods. Just like results on SURF features, ACE is also the best of all the methods, even compared with deep DA methods. The average accuracy of ACE is 92.5%, gaining a 1.8% improvement compared with the best baseline ARTL.

On Office + Home dataset, the performance of ACE is also remarkable. ACE achieves the best performance on 9 out of 12 tasks compared with other baselines. The average accuracy of ACE is 46.2%, gaining a 1.3% improvement compared with the best deep DA method DANN. Excellent performance on deep features reveals that ACE can cooperate with the deep neural networks very well, which greatly increases the applicability of ACE.

4.3. Runtime Analysis

We report the average runtime (training) of six methods on all benchmarks in Table 5. All of the tests are executed using Matlab R2017a on a PC with Intel Core I5 7500 CPU and 16GB DDR4 RAM. For each result in Table 5, we run the algorithms for 10 times and compute the average time. From Table 5, we can get the following observations: 1) Compared with the iterative methods, i.e., JDA and JGSA, ACE not only save plenty of time but also get the higher performance; 2) Compared with the one-step methods, i.e., CORAL, TCA and ARTL, ACE get a significant improvement at the cost of a little more runtime. 3) ACE has an obvious advantage on large-scale datasets. Notably, on high-dimensional features, i.e., Office + Caltech (DeCAF$_6$) and Office + Home, the runtime of CORAL is longer than ACE. CORAL conducts m-dimensional matrix operation, while ACE mainly runs d-dimensional ($d << m$) matrix operation after the eigen-decomposition step. Therefore, an exquisite balance is reached in ACE between training time and performance, which shows the applicability of ACE on large-scale datasets.

4.4. Parameter Sensitivity

Hyper-parameters evaluated are ρ, η, d and λ. Experimental results are shown in Fig. 1.

In Fig. 1(a), when d varying from 60 to 200, accuracies fluctuate in a very small range. In Fig. 1(b), the best value of ρ varies in different evaluations. For DeCAF$_6$, the optimal ρ appears between 25 and 500. For SURF and Office + Home, ρ should be chosen between 1 and 10. In Fig. 1(c), when $\eta \to 0$, all of the accuracies reduce dramatically. When $\eta > 1$, accuracies also reduce. Therefore, the optimal range of η is $[0.1, 1]$. In Fig. 1(d), when $\lambda \to 0$, accuracies are unsatisfactory. λ influences performance little when $\lambda > 1e-6$. When $\lambda > 0.01$, evaluations on all of the datasets can get appealing performance.

5. CONCLUSION

In this paper, we propose a novel method for unsupervised DA, referred to as Adaptive Component Embedding (ACE). ACE aligns the marginal distribution of the source and target domain in a domain-invariant subspace. At the same time, the geometric structures underlying the data manifold are preserved in the process of dimensionality reduction. Moreover, the objective function of ACE can be solved in a closed form, and the training process does not require multiple iterations. Extensive experiments on three benchmarks verify both the

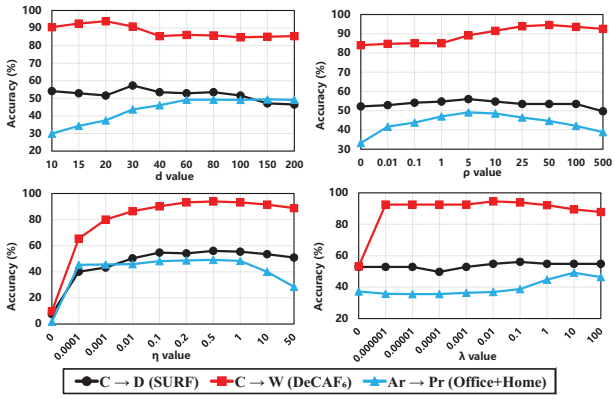

Fig. 1: Parameter sensitivity.

effectiveness and efficiency of ACE.

6. ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61806039, in part by the National Postdoctoral Program for Innovative Talents under Grant BX201700045, in part by the China Postdoctoral Science Foundation under Grant 2017M623006, and in part by Fundamental Research Funds for the Central Universities under No. ZYGX2016J089.

7. REFERENCES

[1] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang, "Domain adaptation via transfer component analysis," *IEEE TNN*, vol. 22, no. 2, pp. 199–210, 2011.

[2] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu, "Transfer feature learning with joint distribution adaptation," in *ICCV*, 2013, pp. 2200–2207.

[3] Baochen Sun, Jiashi Feng, and Kate Saenko, "Return of frustratingly easy domain adaptation.," in *AAAI*, 2016, vol. 6, p. 8.

[4] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang, "From zero-shot learning to cold-start recommendation," in *AAAI*, 2019.

[5] Jing Zhang, Wanqing Li, and Philip Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," *CVPR*, 2017.

[6] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[7] Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan, "Transferable representation learning with deep adaptation networks," *IEEE TPAMI*, 2018.

[8] Jingjing Li, Ke Lu, Zi Huang, Lei Zhu, and Heng Tao Shen, "Transfer independently together: a generalized framework for domain adaptation," *IEEE TCYB*, vol. 49, pp. 2144–2155, 2019.

[9] Jingjing Li, Ke Lu, Zi Huang, Lei Zhu, and Heng Tao Shen, "Heterogeneous domain adaptation through progressive alignment," *IEEE TNNLS*, 2018.

[10] Mingsheng Long, Jianmin Wang, Guiguang Ding, Sinno Jialin Pan, and S Yu Philip, "Adaptation regularization: A general framework for transfer learning," *IEEE TKDE*, vol. 26, no. 5, pp. 1076–1089, 2014.

[11] Jingjing Li, Jidong Zhao, and Ke Lu, "Joint feature selection and structure preservation for domain adaptation.," in *IJCAI*, 2016, pp. 1697–1703.

[12] Jingjing Li, Yue Wu, Jidong Zhao, and Ke Lu, "Low-rank discriminant embedding for multiview learning," *IEEE TCYB*, vol. 47, no. 11, pp. 3516–3529, 2017.

[13] Mengmeng Jing, Jingjing Li, Jidong Zhao, and Ke Lu, "Learning distribution-matched landmarks for unsupervised domain adaptation," in *DASFAA*. Springer, 2018, pp. 491–508.

[14] Liang Xie, Jialie Shen, Lei Zhu, et al., "Online cross-modal hashing for web image retrieval.," in *AAAI*, 2016, pp. 294–300.

[15] Vladimir N Vapnic, "Statistical learning theory," *A Wiley-Interscience Publication*, 1998.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[17] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *CVPR*, 2017, pp. 5018–5027.

[18] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola, "A kernel method for the two-sample-problem," in *NIPS*, 2007, pp. 513–520.

[19] Ulrike Von Luxburg, Mikhail Belkin, and Olivier Bousquet, "Consistency of spectral clustering," *The Annals of Statistics*, pp. 555–586, 2008.

[20] Grace Wahba, *Spline models for observational data*, vol. 59, Siam, 1990.

[21] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *JMLR*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[22] Don Coppersmith and Shmuel Winograd, "Matrix multiplication via arithmetic progressions," in *ACM STOC*, 1987, pp. 1–6.

[23] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell, "Adapting visual category models to new domains," *ECCV*, pp. 213–226, 2010.

[24] Gregory Griffin, Alex Holub, and Pietro Perona, "Caltech-256 object category dataset," 2007.

[25] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014, pp. 647–655.

[26] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.