Theoretic Analysis and Extremely Easy Algorithms for Domain Adaptive Feature Learning

Wenhao Jiang 1∗, Cheng Deng 2, Wei Liu 1, Feiping Nie 3, Fu-lai Chung 4, Heng Huang 5, 2

1 Tencent AI Lab, P. R. China
2 Xidian University, P. R. China
3 Northwestern Polytechnical University, P. R. China
4 Hong Kong Polytechnic University, Hong Kong
5 University of Texas at Arlington, USA

Abstract

Domain adaptation problems arise in a variety of applications, where a training dataset from the source domain and a test dataset from the target domain typically follow different distributions. The primary difficulty in designing effective learning models to solve such problems lies in how to bridge the gap between the source and target distributions. In this paper, we provide comprehensive analysis of feature learning algorithms used in conjunction with linear classifiers for domain adaptation. Our analysis shows that in order to achieve good adaptation performance, the moments of the source domain distribution and target domain distribution should be similar. Based on our new analysis, a novel extremely easy feature learning algorithm for domain adaptation is proposed. Furthermore, our algorithm is extended by leveraging multiple layers, leading to a deep linear model. We evaluate the effectiveness of the proposed algorithms in terms of domain adaptation tasks on the Amazon review dataset and the spam dataset from the ECML/PKDD 2006 discovery challenge.

1 Introduction

In traditional supervised learning, it is assumed that training data samples and test data samples are drawn from the same distribution when a learner (e.g., classifier) is trained. However, this may not be valid in domain adaptation problems. We often have plenty of labeled examples from one domain (source domain) for training a classifier and intend to apply the trained classifier to another different domain (target domain) with very few or even no labeled example. Domain adaptation has tight relationship with semi-supervised learning [Liu et al., 2012] and has been studied before under different names, including covariate shift [Shimodaira, 2000] and sample selection bias [Heckman, 1979; Zadrozny, 2004].

The difficulty of domain adaptation comes from the gap between the source distribution and target distribution. Hence, the learning model trained in the source domain could not be directly used in the target domain. Many algorithms have been proposed to address domain adaptation problems. For example, a semi-supervised method, namely structural correspondence learning (SCL), was proposed in [Blitzer et al., 2006]. SCL defines pivot features which are common to both domains and tries to find the correlation between pivot and non-pivot features. It extracts the corresponding subspace and augments the original feature space with the subspace towards more effective classification. The method in [Blitzer et al., 2011] tries to find the correlations between original features and identify the subspaces in which a good predictor could be obtained by training a classifier merely on the samples from the source domain. The kernel mean matching (KMM) method was proposed in [Huang et al., 2007], which aims to minimize the distance between means of the training and test samples in a reproducing kernel Hilbert space (RKHS) by reweighting the samples from the source domain. In [Chen et al., 2011b], a co-training method called CODA for domain adaptation was proposed. In each iteration, CODA formulates an individual optimization problem which simultaneously learns a target predictor, a split of the feature space into views, and a subset of source and target features to be included in the predictor. It tries to progressively bridge the gap between the source and target domains by adding both the features and instances that the current predictor is most confident about. Nonnegative matrix factorization was employed to bridge the gap between domains by sharing feature clusters [Wang et al., 2011]. Glorot et al. [Glorot et al., 2011] proposed to learn robust feature representations with stacked denoising auto-encoders (SDA) for domain adaptation. Marginalized stacked denoising auto-encoder (mSDA) [Chen et al., 2012], a variant of SDA with a slightly different network structure, was proposed to address the drawback of SDA being too slow in training. Chen et al. [Chen et al., 2012] noticed that the random feature corruption for SDA can be marginalized out, which is equivalent to training a learning model with an infinitely large number of corrupted input data conceptually. Moreover, the linear denoising auto-encoders used in SDA have a closed form, which help speed up the computation. Promising performance was achieved in cross-domain sentiment analysis tasks. In [Sun et al., 2015], correlation alignment (CORAL) exploits the difference between the covariance matrices as a measure of the gap between the source and target domains. A shallow linear
feature algorithm was then proposed. Inspired by the theory of domain adaptation [Ben-david et al., 2006], domain-adversarial neural networks (DANN) [Ajakan et al., 2014] and domain adaptation by back-propagation (DAB) [Ganin and Lempitsky, 2015] optimize an approximated domain distance and an empirical training error on the source domain to seek the hidden representations for both source and target domain samples. DANN and DAB can be treated as classifiers for tackling domain adaptation problems, and also be trained on top of other feature learning algorithms, e.g., mSDA.

Besides the aforementioned methods, domain adaptation has also been studied theoretically. Most of them are built on distances measuring the dissimilarity between different distributions, and generalization bounds are derived based on the proposed distances. In [Ben-david et al., 2006], the A-distance was used to analyze representations for domain adaptation, and VC-dimension shaped generalization bounds were derived for domain adaptation. The analysis showed that a good feature learning algorithm should achieve a low training error on source domain and a small A-distance simultaneously. In [Blitzer et al., 2008], a uniform convergence bound was proved and it was extended to domain adaptation with multiple source domains combined with weights. In [Mansour et al., 2009; Zhang et al., 2013], the A-distance was extended to a more general form and could be used to compare the distance for more general tasks such as regression. All the distances mentioned above were defined in a worst-case sense. In [Germain et al., 2013], a distance defined in an average sense by making use of PAC-Bayesian theory was suggested and an algorithm that simultaneously optimizes the error on the source domain, the hypothesis complexity, and the distance was proposed. In [Ben-David et al., 2010], the conditions for the success of domain adaptation were analyzed. It was shown that a small distance between source and target domains and the existence of a low error classifier on both domains in the hypothesis class are necessary for the success of domain adaptation.

In this paper, we provide theoretic analysis of several key issues pertaining to effective domain adaptive feature learning, showing that the difference (measured by Frobenius norm) between the second moments of source and target domain distributions should be small. Based on this analysis, we propose a simple yet effective feature learning algorithm used in conjunction with linear classifiers. To further improve the feature learning quality, we employ a deep learning approach inspired by stacked denoising auto-encoders in [Glorot et al., 2011; Chen et al., 2012], leading to a deep linear model (DLM). DLMs are easy to analyze and usually are the starting point for theoretical development of neural networks [Goodfellow et al., 2016]. Finally, we demonstrate the effectiveness of our proposed feature learning algorithms on the Amazon review and spam datasets.

2 Theoretic Analysis and New Algorithms

2.1 Notations and Background

Usually, a domain is considered as a pair consisting of a distribution \( \mathcal{D} \) on \( \mathcal{X} \) and a labeling function \( f : \mathcal{X} \rightarrow \{0, 1\} \). In this paper, we consider two domains, a source domain \( (\mathcal{D}_S, f_S) \) and a target domain \( (\mathcal{D}_T, f_T) \). The probability density functions for source distribution and target distribution are \( p_S(x) \) and \( p_T(x) \), respectively. There exist \( n_s \) samples \( \mathcal{D}_S = \{x_1, \ldots, x_{n_s}\} \in \mathcal{R}^d \) with labels \( \mathcal{L}_S = \{y_1, \ldots, y_{n_s}\} \) sampled from the source domain while \( n_t \) samples \( \mathcal{D}_T = \{x_{n_s+1}, \ldots, x_{n_s+n_t}\} \in \mathcal{R}^d \) from the target domain are sampled without labels. They have the same number of features, which is denoted as \( d \). The samples from source domain form the data matrix \( X_S \in \mathcal{R}^{d \times n_s} \) and samples from target domain form \( X_T \in \mathcal{R}^{d \times n_t} \). We use \( X \in \mathcal{R}^{d \times n_s} \) to denote the data matrix containing the samples from both domains.

A hypothesis is a function \( h : \mathcal{X} \rightarrow \{0, 1\} \). The risk of a hypothesis \( h \) over domain \( (\mathcal{D}, f) \) is denoted as \( \epsilon(h, f) = E[|h(x) - f(x)|] \), which is the difference between the two functions. We use notations \( \epsilon_S(h, f_S) = E_S[|h(x) - f_S(x)|] \) and \( \epsilon_T(h, f_T) = E_T[|h(x) - f_T(x)|] \) to denote the risk of \( h \) on source domain and target domain respectively. In this paper, we consider only linear classifiers. We denote the best linear classifier for source domain as \( h_S^* \) with corresponding parameter vector \( w_S^* \), the best classifier for target domain as \( h_T^* \) with parameter vector \( w_T^* \). For simplicity of expression, we define the difference of expectation of a function \( t(x) \) on source domain and target domain as \( E_{S-T}(t(x)) = E_S(t(x)) - E_T(t(x)) = \int t(x)p_S(x) - p_T(x)dx \).

Our goal in this paper is to learn a new feature representation with samples from source domain and target domain such that we can train a classifier with the learned representations on source domain and apply it directly to target domain for achieving low \( \epsilon_T(h, f_T) \) value.

2.2 Analysis of Feature Learning for Linear Classifiers

**Assumption.** We assume that \(|x| \leq \alpha \) for all samples and there exists a low-risk linear classifier \( h^* \) with parameter \( w^* \) for both domains. Its risk on source domain is \( \lambda_1 = \epsilon_S(h^*, f_S) \), and risk on target domain is \( \lambda_2 = \epsilon_T(h^*, f_T) \) and \( \lambda = \epsilon_S(h^*, f_S) + \epsilon_T(h^*, f_T) \), where \( \lambda \) is assumed to be small.

With triangle inequality, it was shown that for any classifier \( h \), the following is satisfied [Blitzer et al., 2008]:

\[
\epsilon_T(h, f_T) \leq \epsilon_T(h^*, f_T) + \epsilon_S(h^*, f_S) + \epsilon_S(h, f_S) + \epsilon_S(h, h^*) + \epsilon_T(h, h^*) \\
= \lambda + \epsilon_S(h, f_S) + \epsilon_S(h, h^*) - \epsilon_T(h, h^*) \\
= \lambda + \epsilon_S(h, f_S) + |E_S[|h(x) - h^*(x)|] - E_T[|h(x) - h^*(x)|]| \\
\]

From this inequality, we can see that the performance of classifier \( h \) on target domain is determined by: 1) the risk of the best classifier \( h^* \) on both domains; 2) the risk of \( h \) on source domain; 3) the difference of dissimilarity between \( h \) and \( h^* \) on both domains. A good feature learning algorithm should decrease the sum of these three terms. In this paper, we focus on the third part and provide an analysis for effective linear feature learning algorithm for linear classifiers.

In the above inequality, \(|h(x) - h^*(x)| \) measures the difference between \( h(x) \) and \( h^*(x) \), which is a non-smooth 0-1 loss and is difficult to analyze. In this paper, we use a smooth approximation for the measure of dissimilarity between \( h \) and
We denote the smooth loss as \( l_{x,y}(z) = l_{x,y}(w^T x) \), where \( w \) is the parameter for the linear classifier \( h(x) \) and \( y \) is the true label of sample \( x \). \( l_{x,y}(z) \) can be logistic loss or smooth approximation of hinge loss [Zhang and Oles, 2001]. We denote \( l_{x,y}^*(z) \) and \( l_{x,y}''(z) \) as the first and second derivatives of \( l_{x,y}(z) \) respectively. Thus, we have the following theorem, which is proved in the supplementary material.

**Theorem 1.** Let \( R = \{x = (x_1, \ldots, x_d)^T : a_i < x_i < b_i (i = 1, 2, \ldots, d)\} \) and \( R' = \{y = (y_1, z)^T = (x_1, \ldots, x_d, z, 1, \ldots, z_d)^T : a_i < x_i < b_i \text{ and } a_i < z_i < b_i (i = 1, 2, \ldots, d)\} \). Assume that any sample in our problem satisfies \( x \in R \), we have the following inequality:

\[
\begin{align*}
&\left| E_S[l_{x,y}^*(w^T x)] - E_T[l_{x,y}^*(w^T X)] \right| \\
&\leq \frac{1}{2} \|w - w^\ast\|^2 \sqrt{CM + \frac{G^2}{M} \|E_{S-T}[xx^T]\|_F^2}
\end{align*}
\]

where \( C, M \) and \( G \) are defined as

\[
\begin{align*}
C &= \frac{1}{2M} \int_{R'} \sum_{i=1}^{2d} \left(|u(y)||D_i u(y)||_\infty + |u(y)||D_i D_i u(y)||_\infty\right) d y, \\
M &= \Pi_{i=1}^{d} (b_i - a_i)^2, \\
u(y) &= l_{x,y}''(x^T w^\ast) l_{x,y}''(z^T w^\ast), \\
D_{1}u(x) \text{ and } D_{2}u(x) \text{ are partial derivatives, } y^\ast \text{ is the label of sample } x \text{ predicted by } w^\ast \text{ as } y^\ast = \text{sgn}(x^T w^\ast).
\end{align*}
\]

Our goal is to find new representations such that the approximation for the measure of dissimilarity between \( h \) and \( h^\ast \) is small. Here, we minimize its upper bound instead. In the right hand side of inequality (1), \( w^\ast \) is the unknown best possible classifier, hence constants like \( C, G \) and \( \|E_{S-T}[xx^T]\|_F^2 \) which are related to \( w^\ast \) could not be optimized directly by feature learning algorithms. We assume that the learned features lie in the same region with the original feature, hence \( M \) could not be optimized. Therefore, in order to achieve good performance, we should minimize \( \|E_{S-T}[xx^T]\|_F^2 \). We will use \( \|E_{S-T}[xx^T]\|_F^2 \) as a measure of domain distance and design a feature learning algorithm based on it.

### 2.3 Algorithm

Let us denote \( M_S = \frac{1}{n_y} X_S X_S^T \), \( M_T = \frac{1}{n_y} X_T X_T^T \) and \( \Delta M = M_S - M_T \), and our goal is to learn a linear transformation \( P \) such that the learned data matrix \( PX \) is suitable for domain adaptation. Based on the above analysis, one of the objectives is to minimize \( \|P^T \Delta M P\|_F^2 \), which is difficult to optimize. Because positive-semidefinite matrices \( A \) and \( B \) satisfy \( 0 \leq \text{tr}(AB) \leq \text{tr}(A) \cdot \text{tr}(B) \), we have \( \text{tr}(\Delta M P P^T \Delta M P^T) \leq \text{tr}(\Delta M P P^T \Delta M) \cdot \text{tr}(P P^T) \). Therefore, the domain distance is bounded by \( \text{tr}(P^T M_2 P) \). Hence, our goal can be expressed as finding a matrix \( P \) with not only small Frobenius norm but also small \( \text{tr}(P^T M_2 P) \). Moreover, the learned representation should also be similar to the original ones. Hence, we can simply use

\[
\min_P \|X^T - X^T P\|_F^2 + \gamma_1 \text{tr}(P P^T) + \gamma_2 \text{tr}(P^T M_2 P),
\]

as objective function\(^1\). In order to force each feature to contribute equally, the length of features should be incorporated by the second term. The final objective function of our method is expressed as follows

\[
\min_P \|X^T - X^T P\|_F^2 + \gamma_1 \text{tr}(P P^T \Lambda P) + \gamma_2 \text{tr}(P^T M_2 P),
\]

where \( \Lambda \) is a diagonal matrix with \( \Lambda_{ii} = X_i X_i^T \), and \( X_i \) is the \( i \)th row of data matrix \( X \). We term our method as **F**eature **L**earning with second **M**oment **M**atching (FLAMM). We will prove that our method will decrease the distance between domains under some conditions. If \( \gamma_2 = 0 \), this algorithm becomes regularized linear regression, which is called simple feature learning algorithm, referred as SFL\(^2\). SFL does not consider minimizing the distance between the second moments explicitly, but it still can improve the adaptation performance in many cases which can be seen from the experimental results section. With our theoretical analysis of domain adaptation, we will make an attempt to illustrate why this simple method works.

To further improve the performance, we adopt the strategy from deep learning methods and apply this algorithm to the learned feature repeatedly. Following the tradition of deep learning, we call the process of finding one linear transformation matrix and updating the data matrix as one layer. The whole algorithm is summarized in Algorithm 1. We can see that the output of our method has the same dimensionality as input. Moreover, the final output is just the linear transformation of the original data matrix, which makes it easier to analyze than the other auto-encoder based methods.

**Algorithm 1** Feature learning with second moment matching (FLAMM).

**Input:** \( X, \gamma_1, \gamma_2 \), number of layers \( K \).

for \( k = 1 \) to \( K \) do

    Compute \( P \) by solving problem (2).

    Update \( X \) by \( X = P^T X \).

end for

**Output:** \( X \)

\(^1\)To express clearly, we omit the bias term.

\(^2\)The linear transformation step of mSDA could be rewritten in a form similar to SFL approximately. Hence, each layer of mSDA could be seen as SFL plus a non-linear transformation step.
domain and target domain. CORAL just whitens source domain and then recolors it with covariance matrix of target domain. The authors did not provide theoretical analysis for the gap measure used and CORAL is also not a deep feature learning method. We will show that our method is better than CORAL.

2.4 Analysis of Our Method

In this subsection, we will provide an analysis of FLAMM. Since FLAMM is just a revision of SFL by adding a term, we will analyze SFL first. We will prove that the domain distance decreases for each layer of SFL under some conditions, which also explains why mSDA works since the layer of mSDA is similar to that of SFL. Based on that, we will provide an analysis of FLAMM. Here are some necessary lemmas.

Lemma 1 (Weyl’s inequality [Horn and Johnson, 2012] Theorem 4.3.1). Let $A, B \in \mathbb{R}^{n \times n}$ be symmetric matrices, and denote the eigenvalue of matrix $A$ as $\lambda_i(A)$, which is arranged in increasing order $\lambda_1(A) \leq \lambda_2(A) \leq \ldots \leq \lambda_n(A)$. For $k = 1, 2, \ldots, n$, we have $\lambda_k(A) + \lambda_1(B) \leq \lambda_k(A + B) \leq \lambda_k(B) + \lambda_1(A)$.

Lemma 2 ([Horn and Johnson, 1991] Theorem 3.3.16). Let $A, B$ be $n \times n$ matrices, and denote the singular values of matrix $A$ as $\sigma_i(A)$, which is arranged in increasing order $\sigma_1(A) \leq \sigma_2(A) \leq \ldots \leq \sigma_n(A)$. For $k = 1, 2, \ldots, n$, we have $\sigma_k(AB) \leq \sigma_k(A)\sigma_n(B)$.

Lemma 3. $P$ is computed by solving problem (2) with $\gamma_2 = 0$, if $\gamma_1 \geq \frac{\lambda_0(XX^T) - \lambda_1(XX^T)}{\lambda_1(XX^T)}$ then $I - PP^T$ is positive definite matrix.

Proof: According to Lemma 1, we have $\lambda_i(XX^T + \gamma_1 A) \geq \gamma_1 \lambda_i(A) + \lambda_1(XX^T)$, therefore $\lambda_i(XX^T + \gamma_1 A)^{-1} \leq \frac{\lambda_1(XX^T)}{\gamma_1 \lambda_i(A) + \lambda_1(XX^T)}$. By Lemma 2, we have:

\[
\sigma_i(P) = \sigma_i((XX^T + \gamma_1 A)^{-1}XX^T) \\
\leq \lambda_i((XX^T + \gamma_1 A)^{-1}) \sigma_i(XX^T) \\
= \frac{\lambda_i(XX^T)}{\gamma_1 \lambda_i(A) + \lambda_1(XX^T)}.
\]

If $\gamma_1 > \frac{\lambda_0(XX^T) - \lambda_1(XX^T)}{\lambda_1(XX^T)}$, we will have $\frac{\lambda_0(XX^T) - \lambda_1(XX^T)}{\lambda_1(XX^T)} < 1$, which means $\sigma_i(P) < 1$. Hence, $\lambda_i(PP^T) = \sigma_i(P)^2 < 1$. Therefore, $I - PP^T$ is positive definite matrix.

Lemma 4. $A, B, C$ are symmetric matrices and $A \succeq 0$, $B \preceq C$, then $\text{tr}(AB) \leq \text{tr}(AC)$.

Proof: If $A$ is a rank-one matrix, we can express $A$ as $A = vv^T$. Then we have $\text{tr}(AB) = \text{tr}(vv^TB) = v^TBv \leq v^TCv = \text{tr}(AC)$. If $A$ is not a rank-one matrix, it can be expressed as $A = \sum_i A_iu_iu_i^T$. Since $A \succeq 0$, denote $v_i = \sqrt{\lambda_i}$, write $A = \sum_i v_i v_i^T$. Then we have

\[
\text{tr}(AB) = \text{tr}(\sum_i v_i v_i^T B) = \sum_i \text{tr}(v_i v_i^T B) \leq \sum_i \text{tr}(v_i v_i^T C) = \text{tr}(AC) \tag{3}
\]

Based on the above lemmas, we have the following theorem:

Theorem 2. If $P$ is computed by solving problem (2) with $\gamma_2 = 0$ and $\gamma_1 \geq \frac{\lambda_0(XX^T) - \lambda_1(XX^T)}{\lambda_1(XX^T)}$, then the inequality $\|\Delta M\|_F^2 \geq \|PP^T \Delta M P^T\|_F^2$ is satisfied.

Proof: In this proof, we will use the following notations:

\[
M = XX^T, \quad M_n = \Lambda^{-\frac{1}{2}} M \Lambda^{-\frac{1}{2}}, \quad \Delta M_n = \Lambda^{-\frac{1}{2}} \Delta M \Lambda^{-\frac{1}{2}}, \quad P_n = \Lambda^{\frac{1}{2}} P \Lambda^{-\frac{1}{2}} = (M_n + \gamma_1 I)^{-1} M_n.
\]

By Lemma 3, we know that if $\gamma_1 \geq \frac{\lambda_0(XX^T) - \lambda_1(XX^T)}{\lambda_1(XX^T)}$ we will obtain that $I - PP^T$ is a positive definite matrix, which means $\Delta = P_n \Lambda P_n^T$ is also a positive definite matrix. Hence, we have $P_n \Lambda P_n^T \succeq I$. Therefore, we obtain:

\[
\frac{1}{F} \|PP^T \Delta M P^T\|_F^2 = \left\| \Lambda^{\frac{1}{2}} P_n \Delta M_n P_n \Lambda^{\frac{1}{2}} \right\|_F^2 \\
= \text{tr}((\Delta M_n P_n \Lambda P_n^T \Delta M_n)(P_n \Lambda P_n^T)) \\
\leq \text{tr}(\Delta M_n P_n \Lambda P_n^T \Delta M_n) \\
= \text{tr}((P_n \Lambda P_n^T)(\Delta M_n \Lambda \Delta M_n)) \\
\leq \text{tr}(\Delta M_n \Lambda \Delta M_n) \\
= \|\Delta M\|_F^2.
\]

From the above theorem, we can see that on a certain layer of SFL, if $\gamma_1$ is big enough distance between the second moments will definitely get smaller compared to that of the input matrix. Hence, SFL will decrease the distance implicitly on one layer. This also explains why mSDA works for domain adaptation problems, since the layer of mSDA is very similar to that of SFL. Therefore, even if SFL and mSDA do not optimize the difference between the two second moment explicitly, they are capable of improving the domain adaptation performance. For FLAMM, with the third term in the objective function, it can decrease the domain distance explicitly. Hence, FLAMM could be seen as a trade-off between reconstruction error and domain distance. And we can achieve the same reconstruction error with smaller domain distance. That’s the reason why FLAMM performs better than SFL, which can be seen in the following section. In practice, we do not need to compute the exact condition in theorem 2, we can just treat $\gamma_1$ and $\gamma_2$ as ordinary parameters and select them using the validation set. We will illustrate the changes of distance empirically in the experimental results section.

3 Experimental Results

3.1 Datasets

We evaluate and analyze the proposed methods on the Amazon review dataset 3 [Blitzer et al., 2007], and the ECML PKDD 2006 spam dataset 4 [Bickel, 2008]. As in [Blitzer et al., 2007], a smaller subset of the Amazon review dataset which contains reviews of four types of products: Books, DVDs, Electronics, and Kitchen appliances, is

\[\text{http://www.cs.jhu.edu/~mdredze/datasets/sentiment/}\]

\[\text{http://www.ecmlpkdd2006.org/challenge.html}\]
Table 1: Domain adaptation performance (accuracy %) on the Amazon review dataset. We use the initial letters B, D, E, and K to represent the four domains. The last line is about the average performance over the 12 tasks.

<table>
<thead>
<tr>
<th></th>
<th>FLAMM₂</th>
<th>SFL₂</th>
<th>FLAMM₁</th>
<th>SFL₁</th>
<th>mSDA₃</th>
<th>tf-idf</th>
<th>PCA</th>
<th>CODA</th>
<th>CORAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>B → D</td>
<td>84.06</td>
<td>84.15</td>
<td>83.31</td>
<td>82.27</td>
<td>83.96</td>
<td>79.00</td>
<td>81.65</td>
<td>81.01</td>
<td>81.46</td>
</tr>
<tr>
<td>B → E</td>
<td>81.07</td>
<td>77.26</td>
<td>79.63</td>
<td>74.97</td>
<td>77.17</td>
<td>73.32</td>
<td>74.83</td>
<td>80.83</td>
<td>74.69</td>
</tr>
<tr>
<td>B → K</td>
<td>83.65</td>
<td>83.01</td>
<td>82.86</td>
<td>82.77</td>
<td>84.11</td>
<td>76.69</td>
<td>81.51</td>
<td>82.63</td>
<td>76.73</td>
</tr>
<tr>
<td>D → B</td>
<td>83.46</td>
<td>83.58</td>
<td>82.14</td>
<td>82.79</td>
<td>83.35</td>
<td>77.12</td>
<td>80.5</td>
<td>79.37</td>
<td>79.57</td>
</tr>
<tr>
<td>D → E</td>
<td>82.69</td>
<td>82.01</td>
<td>80.96</td>
<td>79.75</td>
<td>82.88</td>
<td>73.73</td>
<td>79.65</td>
<td>81.68</td>
<td>76.45</td>
</tr>
<tr>
<td>D → K</td>
<td>88.10</td>
<td>88.03</td>
<td>86.85</td>
<td>86.96</td>
<td>87.49</td>
<td>77.94</td>
<td>81.56</td>
<td>84.24</td>
<td>79.37</td>
</tr>
<tr>
<td>E → B</td>
<td>80.33</td>
<td>79.62</td>
<td>79.31</td>
<td>78.16</td>
<td>80.20</td>
<td>71.34</td>
<td>78.48</td>
<td>77.53</td>
<td>73.36</td>
</tr>
<tr>
<td>E → D</td>
<td>80.56</td>
<td>80.27</td>
<td>79.94</td>
<td>80.04</td>
<td>79.71</td>
<td>71.77</td>
<td>77.77</td>
<td>77.56</td>
<td>74.36</td>
</tr>
<tr>
<td>E → K</td>
<td>89.16</td>
<td>88.56</td>
<td>88.44</td>
<td>88.14</td>
<td>88.43</td>
<td>84.02</td>
<td>86.72</td>
<td>85.89</td>
<td>86.18</td>
</tr>
<tr>
<td>K → B</td>
<td>79.02</td>
<td>79.90</td>
<td>78.21</td>
<td>76.69</td>
<td>71.63</td>
<td>71.77</td>
<td>74.75</td>
<td>76.49</td>
<td>72.66</td>
</tr>
<tr>
<td>K → D</td>
<td>80.82</td>
<td>80.53</td>
<td>79.58</td>
<td>78.03</td>
<td>78.22</td>
<td>73.00</td>
<td>78.02</td>
<td>78.61</td>
<td>74.82</td>
</tr>
<tr>
<td>K → E</td>
<td>86.57</td>
<td>86.67</td>
<td>86.77</td>
<td>86.88</td>
<td>86.91</td>
<td>83.65</td>
<td>85.35</td>
<td>85.59</td>
<td>85.75</td>
</tr>
<tr>
<td><strong>AVG</strong></td>
<td><strong>83.29</strong></td>
<td><strong>82.79</strong></td>
<td><strong>82.34</strong></td>
<td><strong>81.46</strong></td>
<td><strong>82.51</strong></td>
<td><strong>76.11</strong></td>
<td><strong>80.06</strong></td>
<td><strong>80.95</strong></td>
<td><strong>77.95</strong></td>
</tr>
</tbody>
</table>

Table 2: Domain adaptation performance (accuracy %) on the spam dataset.

<table>
<thead>
<tr>
<th></th>
<th>FLAMM₂</th>
<th>SFL₂</th>
<th>FLAMM₁</th>
<th>SFL₁</th>
<th>mSDA₃</th>
<th>tf-idf</th>
<th>PCA</th>
<th>CODA</th>
<th>CORAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public → U₀</td>
<td>91.99</td>
<td>87.09</td>
<td>79.79</td>
<td>65.48</td>
<td>90.74</td>
<td>61.73</td>
<td>66.58</td>
<td>90.45</td>
<td>77.58</td>
</tr>
<tr>
<td>Public → U₁</td>
<td>94.19</td>
<td>93.94</td>
<td>77.88</td>
<td>73.12</td>
<td>92.64</td>
<td>70.67</td>
<td>73.47</td>
<td>92.95</td>
<td>80.98</td>
</tr>
<tr>
<td>Public → U₂</td>
<td>95.15</td>
<td>86.90</td>
<td>85.20</td>
<td>76.75</td>
<td>93.8</td>
<td>74.15</td>
<td>75.65</td>
<td>95.33</td>
<td>87</td>
</tr>
<tr>
<td><strong>AVG</strong></td>
<td><strong>93.78</strong></td>
<td><strong>89.31</strong></td>
<td><strong>80.96</strong></td>
<td><strong>71.78</strong></td>
<td><strong>92.39</strong></td>
<td><strong>68.83</strong></td>
<td><strong>71.9</strong></td>
<td><strong>92.91</strong></td>
<td><strong>81.85</strong></td>
</tr>
</tbody>
</table>

We report the results of two baseline representations. The first one is just the raw tf-idf representation and the second one is the PCA representation. For PCA, the subspace is obtained from both source domain samples and target domain samples. Besides these two baselines, we also compare our method with CORAL [Sun et al., 2015] and CODA [Chen et al., 2011a]. CORAL is an effective shallow feature learning method and has been shown outperform many well-known methods, e.g., GFK [Gong et al., 2012], SVMA [Duan et al., 2012] and SCL [Blitzer et al., 2006], mainly on image domain adaptation tasks. CODA is a state-of-the-art domain adaptation algorithm based on co-training. At last, we compare our method with deep learning methods. Since mSDA is better than SDA as shown in [Chen et al., 2012], we only provide comparisons with mSDA. We use binary representations for mSDA as in [Chen et al., 2012], and for the other learning methods, samples are represented with tf-idf and normalized to have unit length. For representations learned by all feature learning methods, we train a linear SVM on the source domain data and test it on the target domain. The performance metric is classification accuracy.

We denote our method with $K$ layers as FLAMM$_{K}$ and similar notations are also used for SFL and mSDA. We set $K$ as 2 for the Amazon review dataset and 5 for the spam dataset. To provide better understanding of our algorithm, we also provide results with only one layer. Parameters for representation learning process were selected based on the validation set. On the Amazon review dataset, both $\gamma_1$ and $\gamma_2$ were selected from $[10, 20, \ldots, 100]$. On the spam dataset, they were selected from $[10, 30, 50, \ldots, 150]$. For mSDA, the noise was selected from $[0.5, 0.6, 0.7, 0.8, 0.9]$ on the Amazon dataset and $[0.9, 0.91, \ldots, 0.99]$ on the spam dataset. For PCA, the reduced dimensionality was selected from

3The method we used was called STV in [Zhong et al., 2010]. And it works quite well in our setting. Transfer cross validation [Zhong et al., 2010] should provide similar results.
Figure 1: Domain adaptation performances with different number of layers on different datasets. The parameters for the two methods are selected on validation set.

Figure 2: Distances between the second moments of source distribution and target distribution through layers on different datasets with different $\gamma_1$ and $\gamma_2$. The distances are computed with the input data matrix of each layer. For layer 1, it is just the difference between the second moments on the original representations.

[50, 100, 150, 200, 250, 300] for both datasets. For CODA\(^6\), we used the same parameters as in [Chen et al., 2011b], except that we set $\gamma = 0.0001$ on the spam dataset. The results of CODA depend on initializations, hence we run CODA 10 times and provide the average accuracies. The results are presented in Tables 1 and 2. We can see that our method achieved the best performance on the two datasets. Results of FLAMM are better than those of SFL, and results with multiple layers are usually better than results with only one layer. We can see that the proposed objective and multi-layer structure did help improve the adaptation performance.

The performances with different number of layers on the two datasets are also plotted in Figure 1. We can see that our method is better than SFL consistently on both datasets. On the Amazon dataset, our method become worse when $K$ is larger than 4. The reason might be that the difference between the learned representations and the original representations is unnecessarily high.

We also present the distance between the second moments of source domain and target domain through layers with different $\gamma_1$ and $\gamma_2$ when $K = 10$ as shown in Figure 2. We can see that the distance decreases through layers for both FLAMM and SFL and the distance of FLAMM is smaller than that of SFL with the same $\gamma_1$. This observation explained why SFL also improved the adaptation performance. For FLAMM, the parameters control how small domain distance we can get. Bigger $\gamma_1$ and $\gamma_2$ will lead to smaller distance between the two second moments. But at the same time, the learned representations will be more different from the original input\(^7\), which might not be suitable for classifiers. Hence, our method could be understood as a trade-off between domain distance and reconstruction error.

4 Conclusions

In this paper, theoretic analysis of the factors that affect the performance of feature learning for domain adaptation was provided. We found that the distance between the second moments of the source domain and target domain should be small, which is important for a linear classifier to generalize well on the target domain. Based on our analysis, an extremely easy yet effective feature learning algorithm was proposed. Furthermore, our algorithm was extended by leveraging multiple layers, leading to a deep linear model. Meanwhile, we also explained why the simple ridge regression based method which does not minimize the gap between source and target distributions can improve the adaptation performance. The experimental results on the Amazon review and spam datasets corroborated the advantages of our proposed feature learning approach.

Acknowledgments

This work was partially supported by the following grants: NSF-IIS 1302675, NSF-IIS 1344152, NSF-DII 1356628 and RGC GRF Grant No. PolyU 152039/14E.

---

\(^6\)Source code: http://www.cse.wustl.edu/~mchen/code/coda.tar

\(^7\)In our implementation, $\gamma_2$ is the product of a number and $n$, where $n$ is the number of samples.
References


