The $\ell_{2,1}$-Norm Stacked Robust Autoencoders for Domain Adaptation

Wenhao Jiang[1], **Hongchang Gao**[1], **Fu-lai Chung**[2], **Heng Huang**[1*]
[1]Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX, USA
[2]Department of Computing, Hong Kong Polytechnic University, Hung Hom, Hong Kong, China
{cswhjiang, hongchanggao}@gmail.com, cskchung@comp.polyu.edu.hk, heng@uta.edu

Abstract

Recently, deep learning methods that employ stacked denoising autoencoders (SDAs) have been successfully applied in domain adaptation. Remarkable performance in multi-domain sentiment analysis datasets has been reported, making deep learning a promising approach to domain adaptation problems. SDAs are distinguished by learning robust data representations for recovering the original features that have been artificially corrupted with noise. The idea has been further exploited to marginalize out the random corruptions by a state-of-the-art method called mSDA. In this paper, a deep learning method for domain adaptation called $\ell_{2,1}$-norm stacked robust autoencoders ($\ell_{2,1}$-SRA) is proposed to learn useful representations for domain adaptation tasks. Each layer of $\ell_{2,1}$-SRA contains two steps: a robust linear reconstruction step which is based on $\ell_{2,1}$ robust regression and a non-linear squashing transformation step. The experimental results demonstrate that the proposed method is very effective in multiple cross domain classification datasets which include Amazon review dataset, spam dataset from ECML/PKDD discovery challenge 2006 and 20 newsgroups dataset.

Introduction

Traditionally, training of a classifier assumes that the training data and the testing data follow the same distributions or they are in the same domain. However, this may not be valid in some situations, and thus posing a need to train classifiers without such an assumption. For example, we might have plenty of labeled samples from one domain (source domain) but very few or even no labeled samples from a different domain (target domain), and want to train a classifier that will do well on both domains. This is the so-called domain adaptation problem which aims to produce a classifier that is trained on the source domain and generalizes well on the target domain (Pan and Yang 2010). Domain adaptation problem is a fundamental problem in machine learning and has been studied before under different names including covariate shift (Shimodaira 2000) and sample selection bias (Heckman 1979; Zadrozny 2004). The term domain adaptation only started recently to attract the interest of researchers (Blitzer, McDonald, and Pereira 2006; Blitzer 2007; Ben-David et al. 2007; Daume III 2007; Blitzer et al. 2008; Daumé and Marcu 2006).

According to whether there exist labeled samples or unlabeled samples from the target domain in the training stage, domain adaptation could be categorized into three types: unsupervised domain adaptation, supervised domain adaptation and semi-supervised domain adaptation (Jiang 2008; Kumar, Saha, and Daume 2010). In unsupervised domain adaptation setting, there are no samples with labels from target domain. While in supervised domain adaptation setting, some amount of labeled samples are available for training. For semi-supervised domain adaptation, models have access to both labeled and unlabeled data in target domain [1]. In this paper, we focus on improving unsupervised domain adaptation performance by feature learning.

Recently, deep learning methods, which can be seen as feature learning methods, has also been applied to unsupervised domain adaptation to learn features such that the gap between the distributions of source domain and target domain is overcomed. Glorot et al. (Glorot, Bordes, and Bengio 2011) proposed to learn robust feature representations with stacked denoising autoencoders (SDAs) and demonstrated the potential of deep learning in obtaining good performance in sentiment analysis across different domains on Amazon review datasets. In (Chen et al. 2012), the authors found that random corruptions could be marginalized out under a specific network structure, and proposed a marginalized SDA (mSDA). The new method is highly effective for domain adaption problems.

In view of the success of deep learning methods for domain adaption, we make an attempt in this paper to propose a new method called the $\ell_{2,1}$-norm stacked robust autoencoder ($\ell_{2,1}$-SRA) to learn effective representations. With learned features, we just train a classifier with samples from source domain and apply the trained classifier directly to the target domain. State-of-the-art performances were achieved.

[*]To whom all correspondence should be addressed. This work was partially supported by NSF-IIS 1117965, NSF-IIS 1302675, NSF-IIS 1344152, NSF-DBI 1356628, NIH R01 AG049371.

[1]For all these three settings, labeled samples from source domain are available.

Related Works

There have been several methods proposed for unsupervised domain adaptation, and here we highlight a few representative ones. As a semi-supervised method, structural learning (Ando and Zhang 2005) was proposed to utilize the information contained in the unlabeled data. It was extended to domain adaptation setting by structural correspondence learning (SCL) (Blitzer, McDonald, and Pereira 2006). SCL defines pivot features which are common to both source domain and target domain and tries to find the correlation between pivot features and non-pivot features. It extracts the corresponding subspace and augments the original feature space with it for more effective classification. The method has been proved useful for natural language processing tasks (Blitzer, McDonald, and Pereira 2006). Coupled subspaces (Blitzer, Foster, and Kakade 2011) is another subspace method for domain adaptation. In this method, samples are divided into multiple views and cross-correlation matrices are computed, from which projection operators for both source domain and target domain could be determined. With these two projection matrices, a linear predictor could be built.

Chen et. al. (Chen, Weinberger, and Blitzer 2011) proposed co-training for domain adaptation (CODA), which is a variant of the co-training method. CODA does not assume two available views. Instead, in each iteration, CODA formulates an individual optimization problem which simultaneously learns a target predictor, a split of the feature space into views (Chen, Weinberger, and Chen 2011), and a subset of source and target features to be included in the predictor. It bridges the gap between source and target domains by slowly adding both the features and instances that the current algorithm is most confident with to the training set.

Glorot et al. (Glorot, Bordes, and Bengio 2011) proposed to learn robust feature representations with stacked denoising autoencoders (SDA) (Vincent et al. 2010) for domain adaptation. Glorot et al. showed that using SDA-learned features with linear SVM classifiers yields recordable performance for the tasks of sentiment analysis across different product domains on Amazon review dataset (Glorot, Bordes, and Bengio 2011). But the method is slow which limits its use in practice. Marginalized stacked denoising autoencoder (mSDA) (Chen et al. 2012), a variant of SDA with slightly different network structure, was proposed to overcome this drawback. Chen et al. (Chen et al. 2012) noticed that the random feature corruption for SDA can be marginalized out and this is equivalent to training the models with an infinitely large number of corrupted input data conceptually. Moreover, its linear denoising autoencoders have a closed form, which help to speed it up. The denoising step is followed by a non-linear step, which is just a hyperbolic tangent function $\tanh(\cdot)$. In (Chen et al. 2012), promising performance is reported for multi-domain sentiment analysis tasks.

Proposed Method

Notation and background

Throughout this paper, the following definitions and notations are used. For a vector v, we denote the ℓ_2-norm of v by $\|v\|_2$. For a matrix $M \in \mathbb{R}^{m\times n}$, we denote the (i,j)-th element by M_{ij}, the i-th row by $M_{i\cdot}$, and the j-th column by $M_{\cdot j}$. Also, we denote $\|M\|_{2,1} = \sum_i \|M_{i\cdot}\|_2$ as the $\ell_{2,1}$-norm of matrix M.

We follow the setup of traditional domain adaptation as follows. Let us assume that the data come from two domains, i.e. source domain S and target domain T. From the source domain, data $D_S = \{x_1, \cdots, x_{n_s}\} \in \mathbb{R}^d$ with labels $L_S = \{y_1, \cdots, y_{n_s}\}$ are sampled, while from the target domain data $D_T = \{x_{n_s+1}, \cdots, x_n\} \in \mathbb{R}^d$ without labels are sampled. The data samples from these two domains form the data matrix $X = (x_1, \cdots, x_n) \in \mathbb{R}^{d\times n}$ and the label vector of source domain is denoted as Y_S. Our goal is to learn a feature representation, on which a classifier can be learned to predict the labels of samples from the target domain T.

Recall that the basic structure of our method is autoencoder. Usually, an autoencoder contains two parts, an *encoder* and a *decoder*. The *encoder* is a parameterized feature extracting function denoted as $f(\cdot)$, which maps the input data vector $x \in \mathbb{R}^d$ to hidden representations $h = f(x)$ where $h \in \mathbb{R}^{d_h}$. The *decoder* $g(\cdot)$ is also a parameterized function which maps the hidden representations back to the input space forming a reconstruction $r = g(h)$. The parameters of the autoencoders are learned to minimize the reconstruction error, measured by a loss function $l(x, r)$.

Denoising autoencoders (DAs) are slightly different from autoencoders. The input is corrupted by adding noise before mapping them into the hidden representations. DAs are trained to reconstruct the original input x from its corrupted input $\tilde{x}$ by minimizing the loss function $l(x, g(f(\tilde{x})))$.

Stacked denoising autoencoders (SDAs) (Vincent et al. 2010) just stacks DAs together by feeding the output of the previous DA into the current DA to learn high-level features. The parameters of SDA are learned layer by layer greedily.

Robust autoencoder for single layer

The difficulty of domain adaptation lies in the fact that different domain might have different specific features. For example, the words *comprehensive* and *detailed* are usually used to describe books but they are seldom used to describe DVDs, electronics and kitchen appliances. Moreover, *comprehensive* and *detailed* are usually good indicators for positive reviews. A classifier trained on book domain which has high weights on the book domain specific words will not perform well on other domains. To overcome this difficulty, we should learn new representations such that there are no such domain specific features. Since these domain specific features only occur in the related domains, their variances are often small compared to those of common words.

In this paper, we approach the problem from a principal component analysis perspective and propose to increase the ratio of the components with larger variance. If the variance of a component is large, it is more possible that it is important for both domains. For example, *excellent* and *satisfied* can be used in the reviews in all the domains. Hence, we should increase the weights for such features. If a component is just informative in only one domain, the corresponding variance will often be smaller than the components that are informative in both domains. Weights of word like *com-*

prehensive and *detailed* should be weakened. Hence, it is reasonable to adjust the weights of components. Additionally, it is also practical to replace the autoencoders in deep learning methods with PCA.

One may know that PCA truncates the components with small singular values, which will cause information deficiency for the next layer. Hence, we can increase the larger variance components in a soft way, i.e. shrinking the eigenvalue r_i of a covariance matrix by a factor $\hat{r}_i = f(r_i)r_i$. If an eigenvalue r_i is large, meaning that the corresponding component is not noise, it should not be deleted and consequently $f(r_i)$ should be close to 1. On the other hand, if r_i is very small, the shrinkage factor $f(r_i)$ should be close to 0 in order to reduce the noise. Since the eigenvalue of covariance matrix is just the square of the corresponding singular value of the centered data matrix, we can carry out such shrinkage on the singular values of centered matrix.

Inspired by ridge regression (Hastie, Tibshirani, and Friedman 2009), one obvious option to implement the required shrinkage is $\hat{s}_i = \frac{s_i^2}{s_i^2+\lambda} s_i$, where $\lambda \geq 0$ and s_i's are singular values of a data matrix that is centered . It could be seen as the result of the following problem which is similar to ridge regression

$$\min_W \|X^T - X^T W\|_F^2 + \lambda \|W\|_F^2, \tag{1}$$

where $X \in \mathbb{R}^{d \times n}$ is the centered data matrix.

It is well known that the square loss is vulnerable to noise and outliers. The occurrence of outliers might largely influence the result of the objective function. To reduce the effect of outliers, we use robust loss to measure reconstruction errors. $\ell_{2,1}$-norm loss was first proposed in (Ding et al. 2006) as a rotational invariant ℓ_1 norm. It has been successfully adopted in robust principal component analysis (Ding et al. 2006) and robust feature selection (Nie et al. 2010) because of its outlier-resistant property.

In this paper, we use the $\ell_{2,1}$-norm as a measure of reconstruction error for our robust autoencoder. The objective function can be written as

$$\min_W \|X^T - X^T W\|_{2,1} + \lambda \|W\|_F^2. \tag{2}$$

However, in this formulation, the variance of feature might be different. In order to enforce equal contribution from each feature, we introduce the feature variance into the regularization term. Hence, the objective function of our robust autoencoder is

$$\min_W \|X^T - X^T W\|_{2,1} + \lambda \operatorname{tr}(W^T \Lambda W), \tag{3}$$

where Λ is a diagonal matrix and $\Lambda_{ii} = X_{i\cdot} X_{i\cdot}^T$. This is a convex problem with a non-smooth loss function and a smooth regularization. By setting the derivative of Equation (3) w.r.t W to zero, we have

$$\sum_i \frac{1}{2\|x_i - W^T x_i\|_2} (-2x_i x_i^T + 2x_i x_i^T W) + 2\lambda \Lambda W = 0.$$

We denote

$$D_{ii} = \frac{1}{2\|x_i - W^T x_i\|_2}, \tag{4}$$

and the above equation becomes

$$\begin{aligned} &\sum_i D_{ii}(-2x_i x_i^T + 2x_i x_i^T W) + 2\lambda \Lambda W \\ =& -2XDX^T + 2XDX^T W + 2\Lambda W = 0. \end{aligned}$$

Note that D is dependent on X and this equation is difficult to solve. However, we can solve it by updating D and W alternatively. If D is given, W could be computed by

$$W = (XDX^T + \lambda \Lambda)^{-1} XDX^T. \tag{5}$$

We can solve the problem (3) by alternatively updating W and D by (4) and (5) until convergence.

Our algorithm usually converges in less than 10 iterations with relative error 10^{-5}. In practice, we can also use the maximum number of iterations as stopping criteria. In the extreme case that the maximum number of iterations is set as 1, our method is just the regularized least squares regression. And it learns useful features as well.

Feature learning with Stacked Robust autoencoder

The output of above process becomes $\hat{X} = W^T X$, which is essentially a linear transformation of input data matrix. To introduce nonlinearity, the output of linear transformation is then fed into hyperbolic tangent function $\tanh(\alpha x)$, where α is a parameter to be determined by users. As in mSDA (Chen et al. 2012), the non-linear step is independent of the autoencoder which is different from other traditional autoencoders (Bengio, Courville, and Vincent 2012). Hence, it is necessary for users to control the intensity of the squashing effect. The algorithm of $\ell_{2,1}$-norm robust autoencoder is summarized in Algorithm 1.

Following the same strategy adopted by other autoencoder based deep learning methods, we also learn the new representations layer by layer greedily. Let us denote the input sample of l^{th} layer by $z^{(l-1)}$ and the original input as $z^{(0)} = x$. The output of l^{th} layer which is also the input of $(l+1)^{th}$ layer could be written as

$$z^{(l)} = \tanh(\alpha W^{(l)T} z^{(l-1)}), \tag{6}$$

where $W^{(l)}$ is the transformation matrix computed with all input samples of layer l. The original input and output from all layers are concatenated to form the final representations. We can use these new representations directly for classification tasks as in traditional supervised learning. The whole feature learning procedure is summarized in Algorithm 2.

Algorithm 1 $\ell_{2,1}$-norm Robust Autoencoder

```
Input: $X, \lambda$.
Initialize $D_{ii} = 1$ for $i = 1, \cdots, d$.
repeat
    Update $W$ by (5).
    Update $D$ by (4).
until convergence
Output: $\tanh(\alpha(W^T X))$
```

```
Algorithm 2 Feature Learning with $\ell_{2,1}$-SRA
    Input: data matrix $X$, $\alpha$, $\lambda$ and number of layers $L$.
    Initialize $H^{(0)} = X$.
    for $i = 1$ to $L$ do
        Compute $H^{(i)}$ by Algorithm 1.
    end for
    Output: Concatenate $H^{(0)}$ and all the learned features to
    form new representations.
```

Our method uses similar architecture to mSDA, but there are some essential differences. First, we use regularization instead of marginalized corruptions. Hence our method is easier to analyze since the loss + regularizer framework is common in statistics. Second, the intensity of non-line transformation could be controlled by parameters. The necessity can be seen from the following section. Last, we use robust loss function to reconstruct inputs. Hence, our method is more roust to noise and the advantages could be seen from the performance comparisons in the next section.

Experimental Results

Datasets

We test and analyze the proposed method on Amazon review dataset [2] (Blitzer, Dredze, and Pereira 2007), ECML/PKDD 2006 spam dataset [3] (Bickel 2008) and 20 newsgroups dataset [4].

For Amazon review dataset, We follow the convention in (Blitzer, Dredze, and Pereira 2007), and use a smaller subset which contains reviews of four types of products: *Books*, *DVDs*, *Electronics*, and *Kitchen appliances* in our experiments. In this smaller dataset, each domain consists of 2000 labeled inputs and approximately 4000 unlabeled ones (varying slightly between domains), and the classes are balanced. In our experiments, we use the 5000 most frequent terms of unigrams and bigrams as features as in (Glorot, Bordes, and Bengio 2011; Chen et al. 2012). For mSDA and $\ell_{2,1}$-SRA, we use the raw binary unigram/bigram features as their inputs as in (Chen et al. 2012). For all other algorithms, we use *tf-idf* representations.

The second dataset is from the ECML/PKDD 2006 discovery challenge. In this dataset, 4000 labeled training samples were collected from publicly available sources (source domain), with half of them are spam and the other half are non-spam. The testing samples were collected from 3 different user inboxes (target domains), each of which consists of 2500 samples. The distributions of samples from source domain and target domain are different since the sources are different. As in Amazon review dataset, we also chose the 5000 most frequent terms as features. In our experiments, three testing samples were deleted as a result of not containing any of these 5000 terms. Hence, we have 7497 testing samples in this dataset.

[2]http://www.cs.jhu.edu/ mdredze/datasets/sentiment/

[3]http://www.ecmlpkdd2006.org/challenge.html

[4]http://qwone.com/~jason/20Newsgroups/

Table 1: Description of data generated from 20 Newsgroups data

Setting	Source Domain	Target Domain
Comp vs. Rec	comp.windows.x rec.sport.hockey	comp.sys.ibm.pc.hardware rec.motorcycles
Comp vs. Sci	comp.windows.x sci.crypt	comp.sys.ibm.pc.hardware sci.med
Comp vs. Talk	comp.windows.x talk.politics.mideast	comp.sys.ibm.pc.hardware talk.politics.guns

The third dataset was generated from the well-known 20 newsgroups dataset. The 20 newsgroups dataset contains 18774 news documents with 61188 features. It is organized in a hierarchical structure which consists of 6 main categories and 20 subcategories. The task is to classify the main categories and we have adopted a setting similar to (Duan, Tsang, and Xu 2012). The four largest main categories (i.e., *comp*, *rec*, *sci*, and *talk*) were chosen for evaluation. For each main category, the largest subcategory was selected as the source domain, while the second largest subcategory was chosen as the target domain. In our experiments, the largest category *comp* is considered as the positive class and one of the three other categories is considered as the negative class for each setting. Since the subcategories are different, the distributions of source domain and target domain are totally different. The settings of this dataset are listed in Table 1.

Since the distributions of source domain and target domain are different, the traditional cross validation can only select parameters that are suitable for source domain. Hence, we simply use a validation set containing a small number of labeled samples selected randomly from target domain to select parameters for feature learning algorithms. After we have the new features, we treat the classification tasks as traditional ones and find the best classifier on source domain and apply it to the target domain. The validation set was not used to select parameters for classification.

Performance comparison

As a baseline, we train a linear SVM (Chang and Lin 2011) on the raw *tf-idf* representation (Salton and Buckley 1988) of the labeled source domain data and test it on the target domain. When learning the new representations, we do not use any samples from other domains. We also present the results of new representation by projecting the whole dataset (samples from both source domain and target domain) onto a low dimensional subspace obtained by PCA. Besides these two baselines, we also compare with CODA (Chen, Weinberger, and Chen 2011), which 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 (Chen et al. 2012), we only provide comparisons with mSDA. The performance metric is classification accuracy.

For representation learning algorithms, we select parameters with validation set. Once we have the new representations, we do not use validation set to select parameters for classification. Instead, we treat it as a traditional supervised

Table 2: Performance (accuracy %) on Amazon review dataset.

	$\ell_{2,1}$-SRA	CODA	PCA	mSDA	*tf-idf*
B → D	**84.09**	81.98	82.16	83.57	81.56
B → E	76.79	**81.64**	75.28	77.09	71.66
B → K	**85.27**	83.21	77.69	84.92	75.49
D → B	**83.88**	79.34	78.64	83.76	79.84
D → E	83.59	80.68	76.47	**85.06**	74.41
D → K	**87.68**	84.74	82.02	87.49	78.41
E → B	**80.55**	75.71	75.97	80.10	72.73
E → D	**80.36**	76.76	77.32	78.81	74.86
E → K	**88.85**	86.43	86.94	88.19	86.02
K → B	79.02	76.89	75.24	**79.14**	72.28
K → D	**79.88**	77.90	77.49	78.52	75.11
K → E	87.31	85.56	85.56	**87.45**	85.24
AVG	**83.11**	80.91	79.23	82.84	77.30

Table 3: Performance (accuracy %) on spam dataset.

	$\ell_{2,1}$-SRA	CODA	PCA	mSDA	*tf-idf*
Public → U0	**93.75**	68.43	63.88	91.64	62.34
Public → U1	**95.55**	81.48	73.62	92.79	70.38
Public → U2	93.15	88.6	73	**93.65**	74.12
AVG	**94.15**	79.51	70.16	92.69	68.95

learning problem and select parameters for classifiers with cross validation only on source domain. With respect to the number of layers, 5 is used for Amazon review dataset, and 3 is used for both spam dataset and 20 newsgroups dataset.

The performances on Amazon review dataset, spam dataset, and 20 newsgroups dataset are shown in Tables 2, 3 and 4 respectively. The best results in each setting have been marked in bold. One may observe that both mSDA and $\ell_{2,1}$-SRA improved the performance a lot compared to the baseline methods. Also, $\ell_{2,1}$-SRA is better than mSDA in quite a consistent manner.

Analysis

There are three parameters in our method: the intensity of non-linear transformation α, the regularizer coefficient λ and the number of layers. We study the effects of these parameters on B $\rightarrow$ D, Public $\rightarrow$ U0 and Comp vs. Rec datasets.

We fixed the number of layers and plotted the accuracies with different values of α and λ in Figure 1. The number of layers are 5, 3 and 3 for B $\rightarrow$ D, Public $\rightarrow$ U0 and Comp vs. Rec datasets respectively. α controls the intensity of the squashing function. If α is large, the squashing will look like a Heaviside step function. On all these datasets, $\alpha = 1$ does not provide satisfactory performance. Small alpha values, say 2 or 3, are good choices for Amazon review dataset and 20 newsgroup dataset. But for spam dataset, large α are needed. In addition, from Figure 1 we can see that the accuracy curves with different λ show similar appearance, and apparently the value of λ is related to the singular values of the data matrix.

We also studied the effect of the number of layers. We

Table 4: Performance (accuracy %) on 20 newsgroups dataset.

	$\ell_{2,1}$-SRA	CODA	PCA	mSDA	*tf-idf*
Comp vs. Rec	**81.92**	79.27	73.43	79.07	72.07
Comp vs. Sci	**93.04**	70.80	75.03	85.61	69.98
Comp vs. Talk	**97.62**	88.18	92.79	96.83	91.85
AVG	**90.86**	79.42	80.42	87.17	77.97

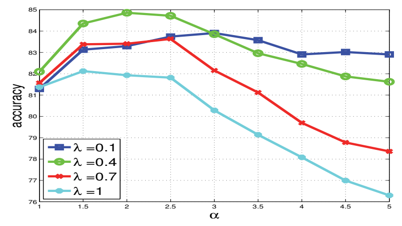
plotted the accuracies with different α, λ and the number of layers on the same datasets as above, which are shown in Figure 2. We can see that our method usually performs the best with $3 \sim 5$ layers. Hence, we use 5, 3 and 3 for Amazon dataset, spam dataset and 20 newsgroup respectively for performance comparisons.

The non-linear transformation step we used does not change the singular values too much. It just squashes the elements to interval $[-1, 1]$. Moreover, we found that even if we deleted the non-linear transformation step, our method still improved the performance. But the performances are not as good as the method with the non-linear squashing step. Moreover, we plotted the eigenvalues of the covariance matrices of output data matrix of each layer in Figure 3. The output of layer 0 is just the original raw data. For each set of eigenvalues that obtained from the same matrix, we normalized it such that it has sum equal to 1. To depict clearly, we only plotted the first few eigenvalues. We can see that bigger eigenvalues become relatively bigger and bigger through layers. The components with smaller eigenvalues become more and more less important through layers. Hence, $\ell_{2,1} - SRA$ can be roughly seen as a procedure of denoising process layer by layer. Through the procedure, the components that are not important for both domains are weakened, the components that are important for both domain are strengthened. We believe that this property is important for $\ell_{2,1} - SRA$ to achieve the best performance.

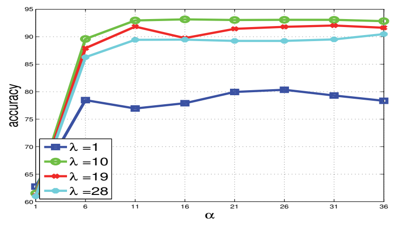
Distance between domains

A-distance is a measure of distance between two distributions. Smaller A-distance means the two distributions are similar to each other. In (Ben-David et al. 2007), it was suggested as a measure of similarity between source domain and target domain. Additionally, it was shown that A-distance is a crucial part of upper bound of generalization for domain adaptation (Ben-David et al. 2007). Hence, the A-distance should be small in order to have good generalization from source domain to target domain. In practice, the exact A-distance is impossible to compute. A proxy-A-distance is used instead and it is defined as $d_A = 2(1 - 2\epsilon)$, where ϵ is the generalization error of a classifier trained to discriminate source domain and target domain. In this paper, ϵ is computed with linear SVM.

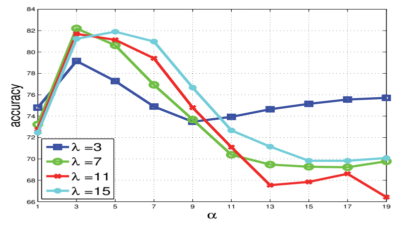
References (Glorot, Bordes, and Bengio 2011) and (Chen et al. 2012) showed that the proxy-A-distance actually increases after the representation learning on Amazon dataset, meaning that the new representations are suitable for both sentiment analysis tasks and domain classification tasks. This phenomenon is also observed on features learned by our method on Amazon review dataset, which can be seen



(a) Amazon review (B → D, 5 layers)

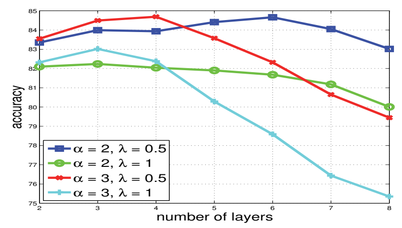


(b) Spam (Public → U0, 3 layers)

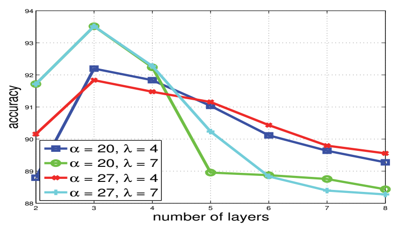


(c) 20 newsgroup (Comp vs. Rec, 3 layers)

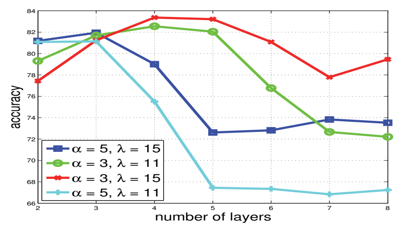
Figure 1: Accuracy (%) with different α and λ on different datasets.



(a) Amazon review (B → D)

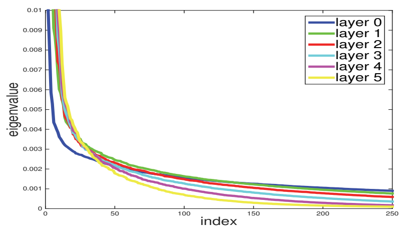


(b) Spam (Public → U0)

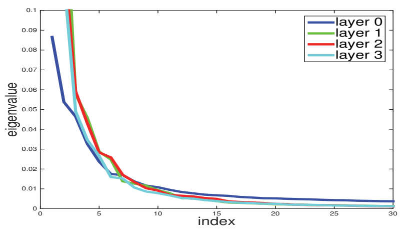


(c) 20 newsgroup (Comp vs. Rec)

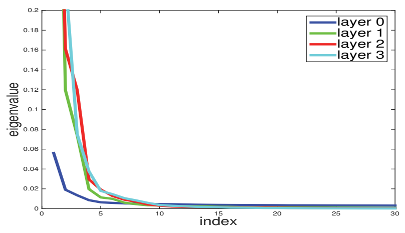
Figure 2: Accuracy (%) with different number of layers on different datasets.



(a) Amazon review (B → D)

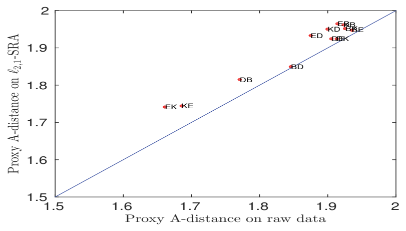


(b) Spam (Public → U0)

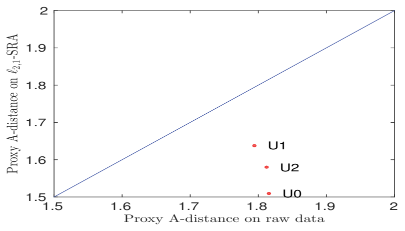


(c) 20 newsgroup (Comp vs. Rec)

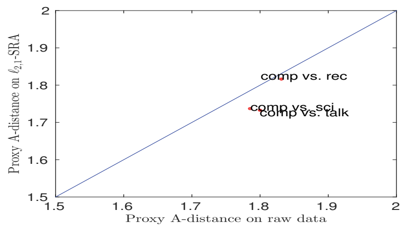
Figure 3: Eigenvalues of covariance matrices of output from each layer on different datasets.



(a) Amazon review



(b) Spam



(c) 20 newsgroups

Figure 4: Proxy-A-distance on different datasets.

from Figure 4(a). But Figure 4(b) and Figure 4(c) show that the proxy-A-distance decreased on spam dataset and 20 newsgroups dataset. However, our method performed well on all the datasets. Therefore, proxy-A-distance is not a necessarily good enough indicator for good domain adaptation performance. The features that are useful for discriminating domain might be weakened or strengthened in the feature learning process. Hence, the proxy-A-distance might get bigger or smaller after feature learning.

Conclusions

The study of deep learning for domain adaptation is still in its preliminary stage. Motivated by the remarkable performance of mSDA (Chen et al. 2012) which is a new deep learning method for domain adaptation, we proposed a $\ell_{2,1}$-norm stacked robust autoencoder ($\ell_{2,1}$-SRA) to learn better representations for domain adaptation tasks, which is a simple combination of statistics tool and deep architecture.

The intuition behind our proposed method is that we should adjust the weights of components of data matrix that compromise of all samples from both source domain and target domain. The proposed autoencoder is based on a loss + regularizer framework, which makes our method easier to implement and analyze compared to neural network based methods. We made an attempt to provide understanding by analyzing the eigenvalues of covariance matrices of output matrix and showed that the noisy components were weakened relatively through layers. The experimental results showed that the proposed method is very promising for domain adaptation tasks.

References

Ando, R., and Zhang, T. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.* 6:1817–1853.

Ben-David, S.; Blitzer, J.; Crammer, K.; and Pereira, F. 2007. Analysis of Representations for Domain Adaptation. In *Advances in Neural Information Processing Systems 19*. 137–144.

Bengio, Y.; Courville, A.; and Vincent, P. 2012. Representation Learning: A Review and New Perspectives. *arXiv preprint arXiv:1206.5538*.

Bickel, S. 2008. ECML-PKDD discovery challenge 2006 overview. In *ECML-PKDD Discovery Challenge Workshop*, 1–9.

Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Wortman, J. 2008. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, 129–136.

Blitzer, J.; Dredze, M.; and Pereira, F. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 440–447.

Blitzer, J.; Foster, D.; and Kakade, S. 2011. Domain adaptation with coupled subspaces. In *Conference on Artificial Intelligence and Statistics, Fort Lauterdale*.

Blitzer, J.; McDonald, R.; and Pereira, F. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, 120–128.

Blitzer, J. 2007. *Domain adaptation of natural language processing systems*. Ph.D. Dissertation, University of Pennsylvania.

Chang, C.-C., and Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2:27:1–27:27.

Chen, M.; Xu, Z.; Sha, F.; and Weinberger, K. Q. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 767–774.

Chen, M.; Weinberger, K. Q.; and Blitzer, J. 2011. Co-Training for Domain Adaptation. In *Advances in Neural Information Processing Systems 24*. 2456–2464.

Chen, M.; Weinberger, K.; and Chen, Y. 2011. Automatic Feature Decomposition for Single View Co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, 953–960.

Daumé, III, H., and Marcu, D. 2006. Domain adaptation for statistical classifiers. *J. Artif. Int. Res.* 26(1):101–126.

Daume III, H. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 256–263. Association for Computational Linguistics.

Ding, C.; Zhou, D.; He, X.; and Zha, H. 2006. R_1-pca: rotational invariant L_1-norm principal component analysis for robust subspace factorization. In *Proceedings of the 23rd international conference on Machine learning*, 281–288. ACM.

Duan, L.; Tsang, I. W.; and Xu, D. 2012. Domain Transfer Multiple Kernel Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 34(3):465–479.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, 513–520.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag New York.

Heckman, J. J. 1979. Sample selection bias as a specification error. *Econometrica: Journal of the econometric society* 153–161.

Jiang, J. 2008. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey*.

Kumar, A.; Saha, A.; and Daume, H. 2010. Co-regularization based semi-supervised domain adaptation. In *Advances in Neural Information Processing Systems 23*. 478–486.

Nie, F.; Huang, H.; Cai, X.; and Ding, C. H. 2010. Efficient and robust feature selection via joint $\ell_{2,1}$-norms minimization. In *Advances in Neural Information Processing Systems*, 1813–1821.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22:1345–1359.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing &amp; Management* 24(5):513 – 523.

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference* 90(2):227–244.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 11:3371–3408.

Zadrozny, B. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, 114. ACM.