126A : Introduction to Data Mining

Professor: Hongfu Liu

Final Project Report

04/25/2020

Kun Li, Ji Tang, Zihao Wang, Han Yue

(Assigned Paper: Return of Frustratingly Easy Domain Adaptation)

We organize this report as follows: Section 1 gives an introduction about the target paper, as well as basic terms for understanding the target paper. Section 2 defines the problem in the target paper, and also introduces the paper's model. In Section 3, we present our model, and how we beat paper's solutions. Section 4 will be our conclusion.

Section 1. Introduction

In this section, we introduce this paper and pre-knowledge for understanding the paper.

The name of the target paper is called "Return of Frustratingly Easy Domain Adaptation", which is published on the AAAI-16 conference website. The authors are Baochen Sun, Jiashi Feng, and Kate Saenko. This target paper focuses on domain adaptation and unsupervised learning.

Before we dive into the target paper, three terms we need to clarify. The first one is "Domain Shift", which means Source data distribution is different from Target data distribution. Take the picture below for example, the bags from Domain 1 do not have any background while those from Domain 2 have a complex background. The classifier trained on Domain one will suffer from performance degradation when it is tested on the other domain.

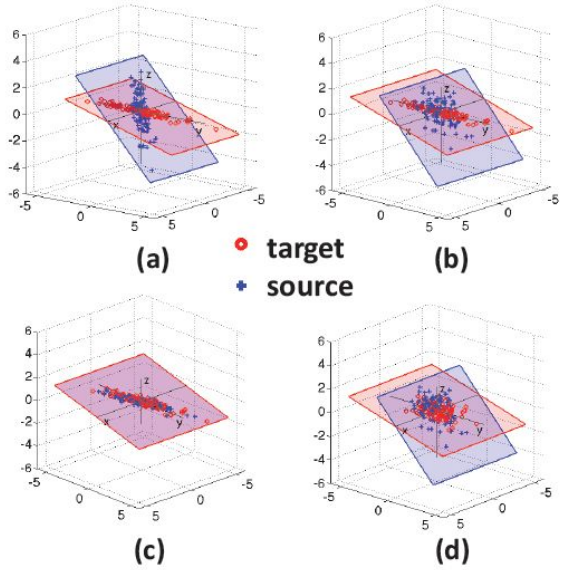
The second term is "Source domain", which is training data distribution. The Last one is "Target domain", which means testing data distribution.

In addition to domain shift, different scenarios can apply. It can be Visual Domain Shift for object recognition, which is the example mentioned above. It also can apply to sentiment prediction, when textual domain shifts.

Section 2. Work of the Assigned Paper

In this section, we clarify the problem definition, the paper's model and we comment on their works.

In the target paper, it discusses an efficient method for unsupervised domain adaptation that minimizes domain shift. For unsupervised domain adaptation, there are several challenges. First, the target domains are unlabeled. During machine learning experiments, there are many scenarios where the target domain is unlabeled, thus requiring unsupervised domain adaptation. Second, subspace projection and hyperparameter selection are expensive, since early techniques for unsupervised adaptation only align the bases of the subspaces, not the distribution of the projected points. They also require expensive subspace projection and hyperparameter selection. The third challenge is that current approaches are limited to two adaptation layers and additional layers can be costly and sensitive to initialization, network structure, and other optimization settings.



In the paper, these steps and challenges for domain adaptation are well defined in this graph. Figure (a) shows the original source and target domains have different distribution covariances, despite the features being normalized to zero mean and unit standard deviation. This presents a problem for transferring classifiers trained on source to target. Figure (b) shows the same two domains after source decorrelation, i.e. removing the feature correlations of the source domain.

Figure (c) shows the target re-correlation, adding the correlation of the target domain to the source features. After this step, the source and target distributions are well aligned and the classifier trained on the adjusted source domain is expected to work well in the target domain. Figure (d) shows that one might instead attempt to align the distributions by whitening both source and target. However, this will fail since the source and target data are likely to lie on different subspaces due to domain shift.

In the target paper, they applied linear transformation A to original source features to minimize the distance between the second-order statistics. The object function is given below where C_S is the covariance of the source features and C_T is the covariance of the target features.

$$\min_A \|A^\top C_S A - C_T\|_F^2$$

It proposes a linear transformation A and the algorithm is shown below:

$$A = U_S E = (U_S \Sigma_S^{+\frac{1}{2}} U_S^\top)(U_{T[1:r]} \Sigma_{T[1:r]}^{\frac{1}{2}} U_{T[1:r]}^\top)$$

The first part whitens the source data by removing the feature correlation of the source domain. And the second part re-colors it with target domains which is adding the correlation of target domain to the source features.

After we understand the algorithm proposed by the paper, we reimplemented it on two datasets. The first one is 12 domain shifts on the Office-Caltech10 dataset (Gong et al. 2012) with SURF features, using the "full training" protocol and the results are shown in table 1.

	A → C	A → D	A → W	C → A	C → D	C → W	D → A	D → C	D → W	W → A	W → C	W → D	AVG
Paper Result	**45.1**	39.5	**44.4**	52.1	**45.9**	**46.4**	**37.7**	**33.8**	**84.7**	**36.0**	**33.7**	**86.6**	**48.8**
Reimplementation	42.5	**40.1**	34.2	**53.2**	45.8	37.9	32.2	29.2	82.8	31.1	30.5	85.4	45.4

Table 1. 12 domain shifts on the Office-Caltech10 dataset (Gong et al. 2012) with SURF features, using the "full training" protocol.

As you can see from the results above, we are able to achieve similar results with the SURF feature. However, compared with other methods, the algorithm proposed by the paper does not have significant improvements on the SURF feature. Then, we also reimplemented the algorithm on extracted deep features which are 6 domain shifts on the Office-Caltech10 dataset (Gong et al. 2012) using the "full training" protocol:

	A --> W	A --> D	D --> A	D --> W	W -->A	W --> D	AVG
Paper Result	61.9	62.2	**48.8**	**96.2**	**48.2**	99.5	69.4
Reimplementation	**65.7**	**64.3**	48.5	96.1	**48.2**	**99.8**	**70.4**

Table 2. 6 domain shifts on the Office-Caltech10 dataset (Gong et al. 2012) with deep features, using the "full training" protocol.

As you can see from the table 2, you can see the results are pretty close and have better results than SURF features. However, we think there is a significant drawback since the paper proposed is not a joint model. It can not generate deep features by itself and needs to rely on extracted features. Therefore, it also shows some limitations to certain datasets that need feature engineering and detect hidden features.
On the other hand, there are also advantages to the target paper. First, it is easy to understand and implement. Second, they achieve high performance on multiple features types and tasks[deep features, CV, NLP]. On the other hand, the disadvantages of the target paper is that they do not have high improvement on SURF features (46.7%). Second, their model is not a joint model. Last, linear transformation may not be a good choice for some datasets.

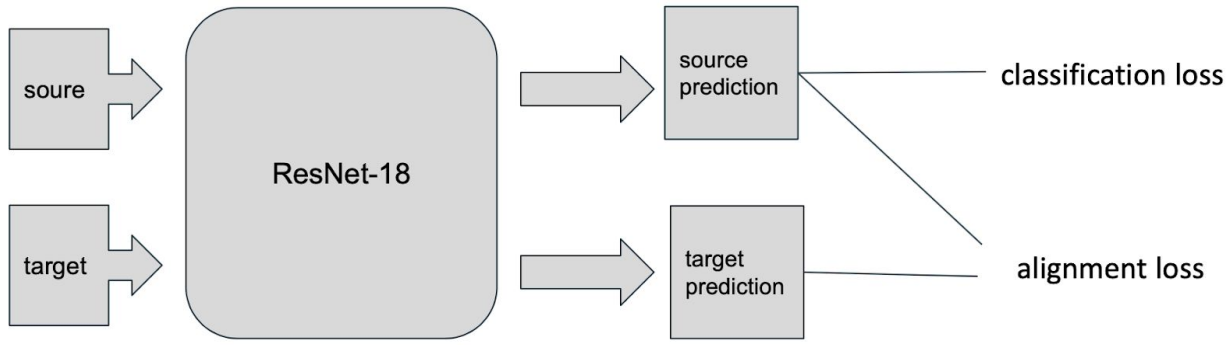
Section 3. Proposed Model

In this section, we introduce our model and how we beat them.

In order to make up for those shortcomings mentioned above, we came up with our own ideas. Basically, we have four ideas. First, we can do deep learning on the transformation process. Instead of using A to transform S → S', we can try a deep learning method to learn nonlinear transformation. Moreover, we can try different distances for correlation alignment. The alignment of CORAL is projected in the Euclidean space, which may not be the best choice for some datasets. Third, the paper just employed the Support Vector Machine (SVM) model to get their results, so we proposed that trying ensemble methods for classification, which is using different models to do majority voting, might be a good idea. Lastly, we can tune more hyperparameters. The paper just tuned C hyperparameter (regularization parameter) in the SVM model, however, we can tune both C and gamma (kernel coefficient) hyperparameters in this model to find the best ones.

Before we dig deeper into our model, we found the source code online and reimplemented it. Our reimplementation result is a little bit better than the paper result, but they are very close. The result is shown in the below table

Our model is focusing on using the nonlinear transformation during the transformation process. Below is the structure of our model:



The inputs would be both features in source and target domain. We put them into a pre-trained ResNet-18 network. And then, the outputs would be source prediction and target prediction. There would be two parts in our total loss. The first part is the classification loss which is used to make sure the source prediction is accurate. The second part is the alignment loss, which is quite similar to the CORAL loss in this paper. It is used to make sure that the source prediction and target prediction are in the same distribution, so that we can do our later classification with the basic assumption met. If you are interested in how our deep learning model works, you can take a look at our codes.

We ran our model in 6 domain shifts on the Office-Caltech10 dataset (Gong et al. 2012), using the "full training" protocol. Table 3 is our result:

	A --> W	A --> D	D --> A	D --> W	W -->A	W --> D	AVG
Paper Result	61.9	62.2	48.8	**96.2**	48.2	**99.5**	69.4
Our Model	**70.2**	**66.06**	**58.8**	93.2	**56.7**	89.0	**72.3**

Table 3. 6 domain shifts on the Office-Caltech10 dataset (Gong et al. 2012), using the "full training" protocol.

Comparing both paper results and our model results, we can see that our model performs much better in most cases. However, our model didn't outperform the paper in D → W and W → D cases. Our explanation is that both D and W are images taken by cameras, so their features might be linear correlated. All in all, the average accuracy of our model is 72.3%, which is around 3% higher than that of the paper. So we can say that we beat this paper!

Section 4. Conclusion

In this section, we make a conclusion about our report including the assigned paper, our model, our result, and comments on our model.

The target paper discusses an efficient method for unsupervised domain adaptation that minimizes domain shift. They applied linear transformation A to original source features to minimize the distance between the second-order statistics. It's very easy to understand and implement with high performance on deep features.

However, the defect of their method is that the linear transformation might not be a good choice for some datasets. That's why we came up with our idea which is to use a nonlinear transformation during the transformation process. In our model, we employed a deep neural network to do the transformation. Even though we didn't outperform the paper on D → W and W → D cases, our result turns out to be 3% higher on average accuracy than the paper did. In this case, we can say that our model is better than the paper's.

However, there are also some shortcomings in our model. For example, we spent hours to train our model. The computation cost is high because our model integrates classification with alignment.