T2S: Domain Adaptation via Model-independent Inverse Mapping and Model Reuse*

Zhi-Yu Shen* and **Ming Li*†**
*National Key Laboratory for Novel Software Technology, Nanjing University
†Collaborative Innovation Center of Novel Software Technology and Industrialization
Nanjing 210023, China
{shenzy, lim}@lamda.nju.edu.cn

***Abstract*—Domain adaptation, which is able to leverage the abundant supervision from the source domain and limited supervision in the target domain to construct a model for the data in the target domain, has drawn significant attentions. Most of the existing domain adaptation methods elaborate to map the information derived from the source domain to the target domain for model construction in the target domain. However, such a 'Source' (S) to 'Target' (T) mapping usually involves 'tailoring' the information from the source domain to fit the target domain, which may lose valuable information in the source domain for model construction. Moreover, such a mapping is usually tightly coupled with the model construction, which is more complex than a separate model construction or mapping construction. In this paper, we provide an alternative way for domain adaptation, named T2S. Instead of mapping the 'S' to 'T' and constructing a model in 'T', we inversely map 'T' to 'S' and reuse the model that has been well-trained with abundant information in 'S' for prediction. Such an approach enjoys the abundant information in source domain for model construction and the simplicity of learning mapping separately with limited supervision in target domain. Experiments on both synthetic and real-world data sets indicate the effectiveness of our framework.**

***Index Terms*—domain adaptation, model reuse**

I. Introduction

Traditional machine learning methods require that the training and testing instances are from the same underlying distribution. However, in many real world tasks, such as sentiment prediction for different types of products and defect prediction for different software projects, training and testing instances may come from different distributions (domains). Such a domain shift may result in the model constructed from one domain no longer be valid for the other domain. To solve this problem, one may need to leverage the data from one domain with abundant supervision (known as the *source domain*) as well as the data from our *target domain* where the prediction is supposed to make to learn a well-performing model for the target domain. Such technique is usually referred as Domain Adaptation [1].

Many domain adaptation approaches have been proposed [4]–[7], [19]–[23], which can be roughly divided into 3 major categories: 1) The first category aims to reuse certain parts of source domain data to improve the performance of the target domain. These useful parts are usually selected by importance sampling and re-weighting techniques. 2) The second category aims to find 'good' feature representations or common feature space to minimize domain divergence. 3) The third category assumes that target domain model should share some parameters or prior distributions of hyper-parameters with the source domain model.

Almost all these approaches elaborate to tailor the source domain data to fit the target domain in the purpose of constructing a well-performing model for the target domain. Although such a solution have been shown its effectiveness for domain adaptation, it may still contain some drawbacks: 1) Insufficient use of source domain information: To well aligning the source domain to the target domain, the source domain is usually tailored to be 'compatible' with the target domain and some information is tailored out during this process. If such information is well-used properly, it may potentially improve the performance of domain adaptation. 2) Mix of domain alignment and model construction: Previous methods usually seeks the mapping from the source to the target domain and constructs the model for the target domain simultaneously in the learning procedure. While to learn a well-performing model requires sufficient information, to learning a mapping as well as the model is more challenging give the information loss by aligning the source to the target domain.

In this paper, we provide an alternative solution to domain adaptation which aims to explicitly address the aforementioned drawbacks by proposing a novel domain adaptation framework called T2S. Instead of mapping the source domain (S) to the target domain (T) and constructing a model in T, T2S inversely maps T back to S and reuses the model that has been well-trained with abundant information in 'S' for prediction. Specifically, T2S consists of two steps: 1) **inversely** mapping the target domain to the source domain (T->S) instead of aligning the source domain to the target domain (S->T); 2) feeding the projections to the pre-trained model of source domain and predicting the labels by the given confidence. These two steps are totally independent of each other.

By such inverse mapping and model reuse, this framework enjoys the following advantages:

- There is abundant supervised information in the source domain while most target data are unlabeled. Therefore, T->S can derive more accurate mapping/alignment than

*This research is supported by National Key Research and Development Program (2017YFB1001903).

S->T because this is aligning the unknown patterns to a 'well-labeled' template.

- Our framework is quite different from the S->T type approaches. When we map the target domain inversely to the source domain, the model construction process can be done separately in the source domain. Since the pre-trained model is independent of the target domain, we can make the most of all the information in the source domain data and any techniques especially those requires huge amount of (supervised) data (e.g. DNNs) to learn a strong model, which consequently may lead to a better prediction performance for target domain prediction if the target domain is well-mapped to source domain.
- Since we only learn the inverse domain mapping instead of the learning whole prediction model simultaneously, the difficulty of learning the mapping is reduced.
- If we need to adapt one source domain to many target domains, we only need to spend efforts in learning a number of mappings for every target domain to the source domain, and reuse the same pre-trained model rather than build different prediction models for different target domains. So this framework can be regarded as a potential technique to realize "Learnware" [2].

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 presents our proposed approach T2S. Section 4 reports the experimental results. Finally, Section 5 concludes the paper.

II. Related Work

Domain adaptation (DA) is a fundamental problem in machine learning [19]–[21] and has a lot of applications in natural language community and computer vision, e.g. [4], [6], [22], [23]. Domain adaptation is regarded as a case of transductive transfer learning. In domain adaptation, the feature spaces between domains are the same, X_S=X_T, but the marginal probability distributions of the data are different, P(X_S)≠P(X_T). In more detail, domain adaptation approaches can be divided into 3 major categories:

1) Instance-selecting based domain adaptation: It assumes that some of the source domain data may be useful in learning for the target domain but some of them may not and could even be harmful. The basic idea of this category is to select the useful part of the source domain. Many researchers have proposed the approaches to achieve this idea. For example, Huang *et al.* [8] proposed a kernel-mean matching algorithm which matches the means between the source domain data and the target domain data, Dai *et al.* [9] extended a traditional Naive Bayesian classifier to solve domain adaptation. Also Dai *et al.* [10] proposed a boosting algorithm which attempts to iteratively re-weight the source domain data to improve the prediction of target domain.

2) Feature-representation based domain adaptation: It assumes that although the distributions of the source domain and target domain are different, there are some latent representations which can represent both two domain uniformly. For example, Blitzer *et al.* [11] proposed a structural correspondence learning (SCL) algorithm to make use of the unlabeled data from the target domain to extract some relevant features that may reduce the difference between the domains and Dai *et al.* [12] proposed a co-clustering based algorithm to propagate the label information across different domains. In [13], Pan *et al.* exploited the Maximum Mean Discrepancy Embedding (MMDE) method to learn a low dimensional space to reduce the difference of distributions between different domains. Long *et al.* [14] proposed an approach for domain adaptation using deep networks which can jointly learn adaptive classifiers and transferable features from labeled data in the source domain and unlabeled data in the target domain.

3) Parameter-sharing based domain adaptation: It assumes that individual models for related tasks should share some parameters or prior distributions of hyper-parameters. The knowledge of source domain can be adapted into target domain by these sharing parameters. There are also many works on this aspect. For example, Lawrence *et al.* [15] proposed an algorithm known as MT-IVM, which is based on Gaussian Processes. Evgeniou *et al.* [16] proposed a method which assumes that the parameter, *w*, in SVMs for each domain can be separated into two terms.

III. Domain Adaptation via Inverse Mapping

The procedure of our domain adaptation framework (T2S) is illustrated in Figure 1 below. When a new target domain instance comes, the first step is to apply mapping functions to it and then get several potential instances(candidates) which can be seen as generated from the source domain for each class. Then in the second step, we predict the candidates' labels using a pre-trained model. For example, in binary classification the model calculates the confidence by $P(x_{pos\_candidate}, label = +1)$ and $P(x_{neg\_candidate}, label = -1)$. At last the predicted label is assigned by the candidate with the higher confidence. In the illustration, the new coming instance is generated from the target distribution of positive class, so the pre-trained model will predict the 'positive candidate' as a positive instance with a higher confidence than the 'negative candidate'. As we could see, inverse mapping is the key step in our framework so the principal issue is how to learn the appropriate mapping functions. The details of the learning procedure are explained in the following subsections.

A. Notations

Before introducing the details of our approach, we first verify the notations used. In this paper, we define the instance of the source domain as x_i^S and the source domain dataset as X^S. In the source domain, the subset of all the positive instances is denoted as X_p^S and the subset of all the negative ones is X_n^S ($X^S = X_p^S \bigcup X_n^S$). Similarly, the instance of the target domain is defined as x_i^T and the target domain dataset is X^T. X^T consists of positive instance subset X_p^T, negative instance subset X_n^T and unlabeled instance subset X_u^T ($X^T = X_p^T \bigcup X_n^T \bigcup X_u^T$ and $|X_p^T| \ll |X_u^T|$, $|X_n^T| \ll |X_u^T|$). We use $F_p(\cdot)$ and $F_n(\cdot)$ respectively to represent the mapping functions for the potential positive and negative instances and

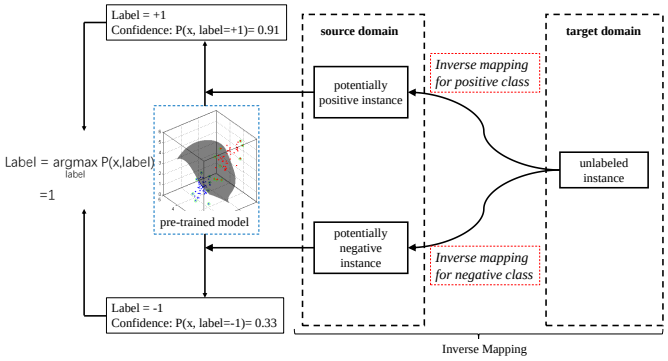

Fig. 1. The general framework of our approach. The right dashed box means that the instance is generated from the target domain distribution. The left dashed box shows that the potential instances follow the source domain distribution. The red dashed boxes are the mapping functions needed to be learned in our approach. The blue dashed box is the pre-trained model.

they could be any functions such as linear mapping or neural network. For two sample sets X_1 and X_2 drawn from two distributions, we define $Dist(X_1, X_2)$ as the distance between these two distributions.

B. Learning the Inverse Mapping

For simplicity, suppose that the problem is to use data in the source domain to help to solve a binary classification problem in the target domain. We have $P_S(x|label = \pm 1) \neq P_T(x|label = \pm 1)$, where $x_i^S \sim P_S(x|label = \pm 1)$ and $x_i^T \sim P_T(x|label = \pm 1)$. In our framework, we first learn two mapping functions which map the target domain distribution to the source domain distribution for each class, which makes $F_p(x_{p_i}^T) \sim P_S(x|label = +1)$ and $F_n(x_{n_i}^T) \sim P_S(x|label = -1)$. Because the domain shift may appear differently for different class, the supervised information can't be fully used if we only use one uniform mapping function. Then for a new coming target instance $x_{u_i}^T$, since we don't know its true label, we map it into the source domain as both a potential positive instance $F_p(X_{u_i}^T)$ and a potential negative instance $F_n(X_{u_i}^T)$. Then we can use a well-performed pre-trained source domain model(M) to classify these two potential instances to obtain the confidence($P(M(F_p(X_{u_i}^T)) = +1|x)$, $P(M(F_n(X_{u_i}^T)) = -1|x)$) and the final label is determined by the potential instance with higher confidence.

From the above, the domain adaptation problem converts to how to learn these mapping functions appropriately. The basic idea of mapping learning process is intuitively to minimize the distance between the source domain distribution and the target domain distribution for each class. We first consider the labeled instances in the target domain. As the only supervised data in the target domain, they should be mapped to the corresponding source domain distribution, so the preliminary objective function is

$$\min_{T_p, T_n} \quad Dist(X_p^S, F_p(X_p^T)) + Dist(X_n^S, F_n(X_n^T)) \tag{1}$$

where $F_p(X) = T_p X$ and $F_n(X) = T_n X$. As we mentioned before, F could be any complex function such as neural network or other nonlinear function. For simplicity, we use linear mapping functions here.

Apart from a few labeled instances in the target domain, most instances in the target domain are unlabeled. Since we don't know their labels, they can be generated from the distribution of any class. So in our approach, we use a parameter w as an indicator to represent the prior probability ($w \in [0, 1]^{N_u}$ is a vector and N_u is the number of unlabeled instances). With this indicator w, we could use the unlabeled data and the mapping functions to minimize the distance between the source domain and the target domain at the same time. Then the formulation becomes

$$\begin{aligned} \min_{T_p, T_n, w} \quad & Dist(X_p^S, F_p(X_p^T)) + Dist(X_n^S, F_n(X_n^T)) \\ & + \alpha \cdot Dist\big(X^S, F_p(X_u^T) \cdot Diag(w) + \\ & F_n(X_u^T) \cdot Diag(\mathbf{1} - w)\big) \\ s.t. \quad & 0 \leq w_i \leq 1 \quad i = 1, 2, ..., N_u \end{aligned} \tag{2}$$

where $\alpha > 0$ is a penalty parameter and $\mathbf{1}$ is an all-one vector. $Diag(w)$ refers to a diagonal matrix whose diagonal elements are the values of w. The added third term measures the whole distance between all the source domain data and all the unlabeled target domain data. In Eq.2, all the target domain instances have been utilized to learn the mapping functions and particularly the supervised information in the target domain is fully used. In the meantime, we don't drop any source domain information. Considering the smoothness in the target feature space and to avoid the overfitting of our approach, we assume that the nearby instances in the target feature space should lead to similar predictive results. To ensure the smoothness of our approach, we finally add a manifold based regularizer (denoted as $f^T L f$) to the objective function [Belkin et al. 2006] where L is the laplacian matrix of training instances. We then derive our final objective function as,

$$\begin{aligned} \min_{T_p, T_n, w} \quad & Dist(X_p^S, F_p(X_p^T)) + Dist(X_n^S, F_n(X_n^T)) \\ & + \alpha \cdot Dist(X^S, F_p(X_u^T) \cdot Diag(w) + \\ & F_n(X_u^T) \cdot Diag(\mathbf{1} - w)) + \beta \cdot f^T L f. \\ s.t. \quad & 0 \leq w_i \leq 1 \quad i = 1, 2, ..., N_u \end{aligned} \tag{3}$$

where $\beta > 0$ is another penalty parameter. Here f is the joint vector of w and the target domain labels $f = [y_1^T, y_2^T, ..., y_n^T, w]^T (n = |X_p^T| + |X_n^T|$ and y_i^T is the label of the labeled target data).

As for criteria function $Dist(\cdot, \cdot)$, we employ the most popular and effective one, Maximum Mean Discrepancy (MMD) [3].

$$\begin{aligned} \min_{T_p, T_n, w} \quad & \text{MMD}(X_p^S, F_p(X_p^T)) + \text{MMD}(X_n^S, F_n(X_n^T)) \\ & + \alpha \cdot \text{MMD}(X^S, F_p(X_u^T) \cdot Diag(w) + \\ & F_n(X_u^T) \cdot Diag(\mathbf{1} - w)) + \beta \cdot f^T L f. \\ s.t. \quad & 0 \leq w_i \leq 1 \quad i = 1, 2, ..., N_u \end{aligned} \tag{4}$$

There are many criteria (such as KL divergence [18]) to measure the distance between two distributions. However, these estimators are parametric or require an intermediate density estimation. MMD is a nonparametric distance estimator which was designed by embedding distributions in an RKHS [25]. MMD is introduced by Gretton *et al.* [26] to compare distributions based on the corresponding RKHS distance. Let's define the the kernel-induced feature map as ϕ. The empirical estimate of MMD between X_1 and X_2 is MMD(X_1, X_2) = $||\frac{1}{n_1}\sum_{i=1}^{n_1}\phi(x_{1_i}) - \frac{1}{n_2}\sum_{i=1}^{n_2}\phi(x_{2_i})||_H^2$ where $||\cdot||_H^2$ is RKHS norm. Therefore, we use the distance between the two mean elements in a RKHS to represent the distance between two distributions. It has been shown by [25] that, when the RKHS is universal, MMD will asymptotically approach zero if and only if the two distributions are the same. Given the criteria function, we can solve the optimization to get the mapping functions $F_p()$ and $F_n()$ and indicator w. The proposed optimization algorithm will be introduced in the next section.

C. Optimization

In Eq.4, instead of finding the nonlinear function ϕ explicitly, we first revisit a dimensionality reduction-based domain adaptation method called MMDE [13]. MMDE embeds both the source and target domain data into a shared low dimensional latent space and then learns the corresponding kernel matrix K by solving a semi-definite programming. It shows that with the kernel trick, the MMD distance can be rewritten as $tr(KQ)$, where K is the kernel matrix defined on all the data, $K_{i,j} = [\phi(x_i)^T\phi(x_j)]$, and Q is a weight matrix, $Q_{i,j} = 1/n_1^2$ if $x_i, x_j \in X$, else $Q_{i,j} = 1/n_2^2$ if $x_i, x_j \in Y$, otherwise, $Q_{i,j} = -(1/n_1 n_2)$. So we convert Eq.4 into the following equivalent form,

$$\begin{aligned} \min_{T_p, T_n, w} \quad & tr(K_{x_p^S, T_p x_p^T} Q_1) + tr(K_{x_n^S, T_n x_n^T} Q_2) \\ & + \alpha\, tr(K_{x^S, F_u(x_u^T)} Q_3) + \beta\, f^T L f. \\ s.t. \quad & 0 \leq w_i \leq 1 \quad i = 1, 2, ..., N_u \\ & F_u(x_{u_i}^T) = w_i T_p x_{u_i}^T + (1 - w_i) T_n x_{u_i}^T \\ & i = 1, 2, ..., N_u \end{aligned} \tag{5}$$

Here, $K_{x_p^S, T_p x_p^T}$,$K_{x_n^S, T_n x_n^T}$ and $K_{x^S, F_u(x_u^T)}$ are the kernel matrices defined on $(X_p^S, F_p(X_p^T))$, $(X_n^S, F_n(X_n^T))$ and $(X^S, F_u(x_{u_i}^T))$. Q_1, Q_2 and Q_3 are the corresponding weight matrices. In this paper we employ alternate descent method to estimate the parameters T_p, T_n and w.

IV. Experiments

In this section we conduct experiments on both synthetic and real data to show the effectiveness of the proposed framework. We select the hyper-parameters of our method as follows. For Gaussian kernel used in *MMD* and the penalty term for unlabeled target domain data, we choose the standard deviation parameter σ and α by cross validation. We only use the labeled source domain data to build a well-performed model as the pre-trained model in our framework.

A. Synthetic Data

We generated the 2-D binary classification source domain and target domain data as Figure 2 shows. As we can see, from the source domain to the target domain, there is a small perturbation for the distribution of positive class. However, for the distribution of negative class, there is a movement and rotation transformation.

Obviously, the distributions of two domains are quite different so we can not reuse the pre-trained source domain model directly. The pre-trained model can not predict the negative target domain instances correctly. Even if we use some existing method to select the appropriate information from the source domain to help the target domain task, we still can't build the "appropriate" model because the distribution of the selected data is still different from that in the target domain essentially. Under our framework, we first learn two mapping functions under which the projection can be regarded as generated from the source distribution. Using our optimization algorithm, we get two mapping functions and Figure 3 shows the mapping results in which the top subfigure shows the positive candidates and the bottom subfigure shows the negative candidates. In Figure 3, we use red 'x' to represent the projection of target instance whose true labels are positive and blue 'x' to represent the projection of target instances whose true labels are negative. Then we could see that the instances from the corresponding class are mapped to the correct region. In our framework these instances can be seen as generated from the corresponding source domain distribution and can be classified by the pre-trained model with high confidence. The black dashed contours in the Figure 3 show the decision region of

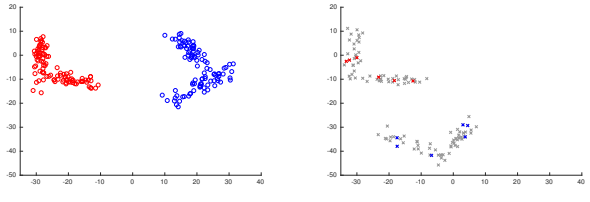

Fig. 2. The synthetic data of source and target domain. The red/blue 'o': positive/negative instance in the source domain. The grey 'x': unlabeled instance in the target domain. The colored 'x': labeled instance in the target domain.

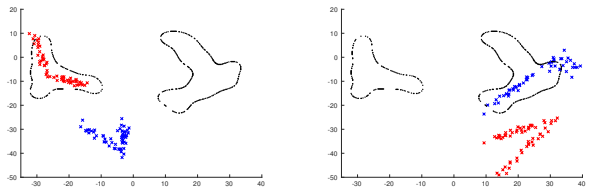

Fig. 3. The result of the mapping functions. The red/blue 'x' represents the projection of target instances whose original labels are positive/negative. The black dashed contours are the decision regions of the pre-trained model with confidence>0.8

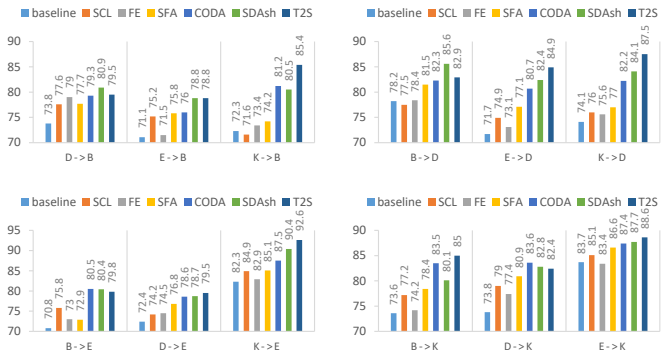

Fig. 4. Accuracy of domain adaptation between all pairs.

the pre-trained model with confidence > 0.8. The result on synthetic data shows that in our framework using a simple pre-trained SVM model could achieve a good performance. The accuracy before inverse mapping is 76.7% and it is 100% after. This experiment also shows that our framework works intuitively and empirically.

B. Real-world Data

We also compare our framework with alternatives on the Multi-Domain Sentiment Dataset introduced in [6]. The Multi-Domain Sentiment Dataset contains product reviews taken from Amazon.com from many product types (domains). Each domain (books, dvds...) has hundreds of thousands of reviews. Each review consists of a rating (0-5 stars), a reviewer name and location, a product name, a review title and date, and the review text. Reviews with rating > 3 were labeled positive, those with rating < 3 were labeled negative, and the rest discarded because their polarity was ambiguous. Each domain contains 1,000 positive and 1,000 negative reviews and thousands unlabeled reviews. We use every domain as the source domain and target domain respectively, so that it generates 12 tasks altogether. In each task, all the instance labels in the source domain and a few instance labels in the target domain are used for learning the mapping functions. Then we evaluate all the methods on the unlabeled instances in the target domain. In this experiment, we choose the parameters by cross validation.

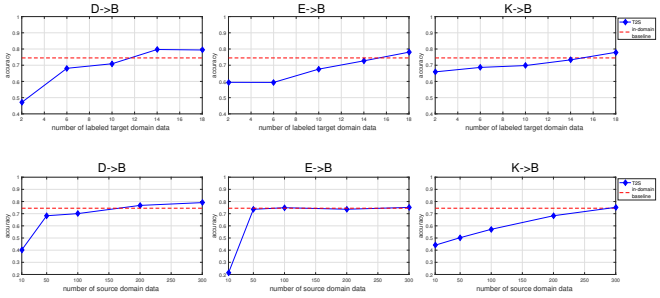

Fig. 5. Accuracy under different number of labeled target data/source domain data.

We choose a simple SVM model trained on the source domain as the baseline. For fair comparison, this model is also used as the pre-trained model in our framework. We compared our framework with SCL [11], FE [17], SFA [4], CODA [27] and SDAsh [5].

- SCL is a structural correspondence learning algorithm. Its key idea is to make the selection of pivot features that are used to link the source and target domains.
- FE is a feature ensemble method for domain adaptation. In FE, we first train individual classifiers with different feature sets divided by their POS tags. The final model is a weighted ensemble of individual classifiers.
- SFA is a spectral feature alignment algorithm to align the domain-specific words from the source and target domains into meaningful clusters, with the help of domain independent words as a bridge.
- CODA is a variant of co-training which bridges the gap between source and target domains by slowly adding to the training set both the target features and instances in which the current algorithm is the most confident.
- SDAsh is a deep learning approach which learns to extract a meaningful representation for each domain in an unsupervised fashion. The basic framework for SDAsh is the Stacked Denoising Auto-encoder.

In the first experiment, we use accuracy as the evaluation criteria. Figure 4 shows the results for all pairs of domain adaptation. The target domains are ordered clockwise from the top left: books, dvd, electronics, and kitchen. For each set of bars, the first letter means the source domain and the second letter means the target domain. The dark blue bars show the performance of our approach. We can see that our approach achieved the best performance in most tasks. The accuracy of our approach is between $78.8\% \sim 92.6\%$. As we could see, when we use "books-reviews" as the target domain, some of the compared methods even have worse performance than the baseline method. Our method achieves a relatively good performance compared with the state-of-the-art methods because we make sufficient use of the labeled data in the target domain and don't destroy the completeness of supervised information in the source domain. What's more, this performance is only achieved by a simple pre-trained SVM model and two simple linear mapping functions, which means that we could get higher performance using more powerful

pre-trained model and more complex mapping functions.

In the second experiment, we also explore the performance under different number of labeled data. As we have mentioned, in our framework we don't need to use the full dataset to learn the domain adaptation model because the inverse mapping is separated from the building model process. We choose the ''books" domain as target domain and every other review data as source domain. We use $2,000$ labeled target domain instances to build a in-domain SVM model for the baseline and use 2000 source domain instances to build an SVM model as the pre-trained model. All the parameters are also chosen by cross validation. The results of the second experiment is:

- We use a fixed number of source domain data and various number (2,5,10,15,20) of target domain data to conduct the experiment. Figure 5 above shows the results of the three pairs.
- We also use different number (10,50,100,200,300) of source domain data and a fixed number of target domain data. Figure 5 below shows the results of the three pairs.

The experiment on the real-world data shows that in our framework we can achieve efficient domain adaptation compared with some state-of-the-art methods and we can also benefit from separating the process of building model and bridging the domain gap.

V. Conclusion

In this paper, we investigate a new framework for domain adaptation. We propose a new approach for domain distribution mapping and model reuse, which is efficient and outperforms other baseline methods. An efficient optimization algorithm is also proposed for learning the mapping functions. A new view of how to solve the domain adaptation problem is given and it may be a probable way to analyze the quality of domain adaptation process. In our framework, we point out that constructing a prediction model and bridge the gap of different domains could be two different tasks and it could not be appropriate to solve them together. We separate the process of constructing model and the process of bridging the gap between different domains, which decreases the difficulty of jointly solving the learning task and bridging the gap of different domains. The inverse mapping makes the bridge from the target domain back to the source domain, which makes it possible to reuse the powerful pre-trained model of the source domain. The experimental results show that we can use only a small part of the data to bridge the gap although we need big data to build the prediction model. The formalization of our approach indicate that we could replace with other powerful mapping function in the future work.

Acknowledgement

The authors would like to thank Dr. Yu-Feng Li for his valuable suggestions to this paper.

References

[1] S.J. Pan and Q. Yang, "A survey on transfer learning," TKDE, vol. 22, pp. 1345–1359, 2010.
[2] Zhi-Hua Zhou, "Learnware: on the future of machine learning,", Frontiers of Computer Science, vol. 10, pp. 589–590, 2016.
[3] Johan A.K. Suykens, "Data visualization and dimensionality reduction using kernel maps with a reference point,", IEEE Transactions on Neural Networks, vol. 19, pp. 1501–1517, 2008.
[4] S.J. Pan and X. Ni and J. Sun and Q. Yang and Z. Chen, "Cross-domain sentiment classification via spectral feature alignment,", ICWWW, pp. 751–760, 2010.
[5] X. Glorot and A. Bordes and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach,", ICML, pp. 513–520, 2011.
[6] J. Blitzer and M. Dredze and F. Pereira, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification,", ACL, pp. 440–447, 2007.
[7] O. Sener and H.O. Song and A. Saxena and S. Savarese, "Learning transferrable representations for unsupervised domain adaptation,", NIPS, pp. 2110–2118, 2016.
[8] J. Huang and et al., "Correcting sample selection bias by unlabeled data,", NIPS, pp. 601–608, 2007.
[9] W. Dai and G.R. Xue and Q. Yang and Y. Yu, "Transferring naive bayes classifiers for text classification,", AAAI, vol. , pp. 540–545, 2007.
[10] W. Dai and Q. Yang and G.R. Xue and Y. Yu, "Boosting for transfer learning,", ICML, pp. 193–200, 2007.
[11] J. Blitzer and R. McDonald and F. Pereira, "Domain adaptation with structural correspondence learning,", EMNLP, pp. 120–128, 2006.
[12] W. Dai and G.R. Xue and Q. Yang and Y. Yu, "Co-clustering based classification for out-of-domain documents,", SIGKDD, pp. 210–219, 2007.
[13] S.J. Pan and J.T. Kwok and Q. Yang, "Transfer Learning via Dimensionality Reduction,", AAAI, pp. 677–682, 2008.
[14] M. Long and H. Zhu and J. Wang and M.I. Jordan, "Unsupervised domain adaptation with residual transfer networks,", NIPS, pp. 136–144, 2016.
[15] N.D. Lawrence and J.C. Platt, "Learning to learn with the informative vector machine,", ICML, pp. 65–72, 2004.
[16] T. Evgeniou and M. Pontil, "Regularized multi–task learning,", SIGKDD, pp. 109–117, 2004.
[17] R. Samdani and W. Yih, "Domain adaptation with ensemble of feature groups,", IJCAI, pp. 1458–1464, 2011.
[18] M. Sugiyama and S. Nakajima and H. Kashima and et al, "Direct importance estimation with model selection and its application to covariate shift adaptation,", NIPS, pp. 1433–1440, 2008.
[19] L. Duan and I.W. Tsang and D. Xu, "Domain transfer multiple kernel learning,", TPAMI, vol. 34, pp. 465–479, 2012.
[20] K. Zhang and B. Schlkopf and K. Muandet and Z. Wang, "Domain adaptation under target and conditional shift,", ICML, pp. 819–827, 2013.
[21] X. Wang and J. Schneider, "Flexible transfer learning under support and model shift,", NIPS, pp. 1898–1906, 2014.
[22] K. Saenko and B. Kulis and M. Fritz and T. Darrell, "Adapting visual category models to new domains,", ECCV, pp. 213–226, 2010.
[23] B. Gong and Y. Shi and F. Sha and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation,", CVPR, pp. 2066–2073, 2012.
[24] S.J. Pan and I.W. Tsang and J.T. Kwok and Q. Yang, "Domain adaptation via transfer component analysis,", IEEE Transactions on Neural Networks, vol. 22, pp. 199–210, 2011.
[25] A. Smola and A. Gretton and L. Song and B. Schlkopf, "A Hilbert space embedding for distributions,", ALT, pp. 13–31, year.
[26] A. Gretton and K.M. Borgwardt and M. Rasch and B. Schölkopf and A.J. Smola, "A kernel method for the two-sample-problem,", NIPS, pp. 513–520, 2007.
[27] Minmin Chen and Kilian Q. Weinberger and John C. Blitzer, "Co-Training for Domain Adaptation,", NIPS, pp. 2456–2464, 2011.