UNSUPERVISED DOMAIN ADAPTATION USING NORMALIZING FLOWS

Medha Agarwal *
Department of Statistics
University of Washington
Seattle, WA 98105, USA
medhaaga@uw.edu

ABSTRACT

This manuscript delves into the challenge of unsupervised domain adaptation for binary classification tasks. In this scenario, labeled data is available from a source domain, while the target domain comprises only unlabeled data, necessitating the learning of a classifier. Leveraging the structured nature of text data through BERT embeddings, we employ transport maps to effectively align target features with those from the source domain. Our approach operates under the assumption that label distribution conditioned on features remains independent of domain shifts. We achieve this by employing an unsupervised learning architecture based on normalizing flows to learn transport maps. Through a comprehensive analysis conducted on the Amazon review dataset Ni et al. (2019), which encompasses diverse product categories aiding domain differentiation, we showcase the efficacy of our method.

1 INTRODUCTION

Unsupervised domain adaptation (UDA) is a popular problem in machine learning research, addressing the critical challenge of transferring knowledge from a source domain, where labeled data is abundant, to a target domain, where labeled data is scarce or entirely absent (Ben-David et al., 2006; Pan & Yang, 2009). In the realm of artificial intelligence, UDA plays a pivotal role in enhancing the adaptability of models to diverse and unseen environments. Unlike supervised learning paradigms, UDA operates without explicit target domain labels during the training phase, relying on the inherent structure of the data to bridge the gap between domains. This field holds immense promise for real-world applications, spam recognition, natural language processing, and various domains where model generalization across heterogeneous data distributions is paramount. As machine learning research delves deeper into developing robust UDA techniques, the potential to improve model performance in novel domains is immense.

We work within the covariate shift assumption where the domains differ by the change in distribution of their features. Whereas, the conditional distributions of the labels remain unchanged. For instance, in medical imaging, a model trained to detect tumors from MRI scans may encounter covariate shift if the distribution of patient demographics (e.g., age, gender) varies between data collected from different hospitals. Similarly, in natural language processing, sentiment analysis models trained on customer reviews from one e-commerce platform may face covariate shift if the language usage and expression patterns differ significantly in reviews from another platform. Addressing covariate shift is crucial for ensuring the robustness and generalization capability of classifiers across diverse real-world scenarios.

1.1 LITERATURE REVIEW

Past work has explored the problem of domain adaptation using various methods including instance reweighting (Mansour et al., 2008; Sugiyama et al., 2007), subsampling (Chen et al., 2011), and kernel alignment (Zhang et al., 2013). In a more general work by Courty et al. (2017), the authors

*https://medhaaga.github.io/

get rid of the assumptions of same conditional distribution of labels across domains and learn an optimal transport map between the joint distribution (of features and labels) of source and target domain. Many works have focused on finding a common latent space where both source and target features can be projected. Since the projections into the latent space are equally distributed for all domains, classifiers are learned using the projections from the source domain Pan et al. (2010); Daumé III (2009). Our method aligns with this class of methods where we learn to transport features form source and target domains to a common latent space distributed according to standard normal distribution.

2 METHOD

2.1 PROBLEM SETUP

We consider a simple task of binary classification in two domains setting - a source domain and a target domain. Let $\mathcal{X}_S \subset \mathbb{R}^d$ denote the space of the source features and $\mathcal{X}_T \subset \mathbb{R}^d$ denote the space of target features. We denote the source distribution on $\mathcal{X}_S$ by $\mathcal{D}_S$ and the target distribution on $\mathcal{X}_T$ by $\mathcal{D}_T$. Let $X_S \sim \mathcal{D}_S$ and Y_S be the binary random variable denoting the source labels and (X_T, Y_T) denote similar random variables for the target domain. We make a simplifying assumption that the conditional distribution of the source and target labels is the same, i.e.

$$Y_S|X_S \stackrel{d}{=} Y_T|X_T. \tag{1}$$

This assumption allows us to transfer a classifier learned on source labeled data to target domain once its features are adjusted to "imitate" the source features. We assume that we have n_S labeled samples from the source distribution $\{(\mathbf{x}_S^i, y_S^i)\}_{i=1}^{n_S}$ and only n_T unlabeled samples from the target denoted by $\{(\mathbf{x}_T^j)\}_{j=1}^{n_T}$. The goal is to predict the target labels $\{\hat{y}_T^j\}_{j=1}^{n_T}$ under the assumption equation 1.

2.2 METHOD

Like we said earlier, assumption 1 motivates *transferring* the target features to the source features so that a classifier learned on source data can be directly used for predicting labels on the target domain. This transfer is done by first transporting both source and target feature distributions - $\mathcal{D}_S$ and $\mathcal{D}_T$ to a standard Gaussian distribution. This transportation is done via *deterministic* diffeomorphisms $f_S : \mathcal{X}_S \to \mathbb{R}^d$ and $f_T : \mathcal{X}_T \to \mathbb{R}^d$ such that

$$f_S(X_S) \sim G \quad \text{and} \quad f_T(X_T) \sim G\,,$$

where G is a standard d-dimensional Gaussian random variable. The machine learning architecture used to learn the transport maps f_S and f_T is normalizing flow Kobyzev et al. (2020). Since the empirically known distributions are transports to a normal distribution, hence the name "normalizing flow". Now notice that because both f_S and f_T are invertible, we an claim that

$$f_S^{-1} \circ f_T(X_T) \sim \mathcal{D}_S\,,$$

allowing use to effectively transfer X_T to X_S. By assumption 1, we have that

$$Y_T|X_T \stackrel{d}{=} Y_S|X_S \stackrel{d}{=} Y_S|f_S^{-1} \circ f_T(X_T)\,.$$

Therefore, the problem of learning a binary classifier on (X_T, Y_T) without observations from Y_T can be split into two parts:

1. Learning a binary classifier on (X_S, Y_S).
2. Learning the transport maps f_S and f_T.

Before we dive into the experimental explorations of this technique, we present a background on normalizing flows in the next section.

3 NORMALIZING FLOWS

Normalizing flows (NFs) are a class of generative models that transform a complex probability distribution into a simple one using a series of invertible and differentiable transformations. The basic idea is to start with observed data from a complex unknown probability distribution, and then use a series of invertible transformations, such as affine transformations or permutation operations, to map the distribution to a simple reference distribution, such as the standard Gaussian distribution. The exact characterization of the unknown density function is then obtained by applying a change of variables to the reference distribution. The advantage of Normalizing Flows is that they can generate from complex distributions without requiring the use of complex latent variables or architectures, as is often the case with other generative models like GANs and VAEs. However, they can be computationally expensive, as each transformation requires the calculation of the Jacobian determinant, which can be challenging for high-dimensional data.

Let X be the random variable with unknown complex density function $p : \mathbb{R}^d \to \mathbb{R}$. Let S be a diffeomorphism, i.e. it is bijective, differentiable, and its inverse is also differentiable. Then using the change of variable formula, we know that the probability density function of the random variable $Y := S(X)$ is defined as

$$q(\mathbf{y}) = p \circ S^{-1}(\mathbf{y}) |\nabla S^{-1}(\mathbf{y})| \,. \tag{2}$$

This new density function q is called a pushforward density of the density p by the function S and denoted by $S_{\#}p$. In the context of normalizing flows, the goal is to find the function S such that the pushforward distribution is the chosen simple reference distribution, like the standard Gaussian distribution. Let g be the density function of d dimensional standard Gaussian distribution. Then the goal of finding S such that $q = g$, if achieved, allows us to use the change of variable formula to exactly characterize the unknown probability density function p as

$$p(\mathbf{x}) = g \circ S(\mathbf{x}) |\nabla S(\mathbf{x})| \,. \tag{3}$$

Traditionally, flows are constructed by composing many non-linear invertible functions (bijections). Such a construction banks on the knowledge that the composition of invertible functions is itself invertible and the determinant of its Jacobian has a specific form. Specifically, if $S_1, \ldots, S_k$ is a set k bijective functions and $S := S_k \circ S_{k-1} \circ \cdots \circ S_1$, then S is also a bijective function with

$$S^{-1} = S_1^{-1} \circ S_2^{-1} \circ \cdots \circ S_k^{-1}$$

and

$$\det \nabla S(\mathbf{x}) = \prod_{i=1}^{k} \det \nabla S_i(\mathbf{x}_i)$$

where $\mathbf{x}_i = S_i \circ \ldots S_1(\mathbf{y})$ is the ith intermediary value. Thus any number of bijective functions can be composed to create increasingly complex normalizing flows. Such S is recovered by solving an optimization problem that aims to minimize the KL divergence between $S_{\#}f$ and g. Let $\mathcal{S}$ be an appropriate set of diffeomorphisms, then the optimization problem is

$$\begin{aligned} &\min \mathcal{D}_{\text{KL}}(S_{\#}f|g) \\ &\text{s.t.} \det \nabla S > 0 \text{ and } S \in \mathcal{S} \,. \end{aligned}$$

The global minimizer of the above optimization problem is such that $\mathcal{D}_{\text{KL}}(S_{\#}f|g) = 0$, i.e. $S_{\#}p = g$. Since $\mathcal{D}_{\text{KL}}(S_{\#}p|g) = \mathcal{D}_{\text{KL}}(f|S_{\#}^{-1}g)$, the population objective can also be written as

$$\begin{aligned} &\min \mathbb{E}_p[-\log g \circ S(X) - \log |\det \nabla S(X)|] \\ &\text{s.t.} \det \nabla S > 0 \text{ and } S \in \mathcal{S} \,. \end{aligned}$$

Since the objective involves taking expectation with respect to the target distribution, we can use its Monte Carlo estimator to approximate the optimization objection. Let $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be i.i.d. samples from p, then the sample average version of the optimization problem is

$$\begin{aligned} &\min -\frac{1}{n} \sum_{i=1}^{n} [\log g \circ S(\mathbf{x}_i) + \log |\det \nabla S(\mathbf{x}_i)|] \\ &\text{s.t.} \det \nabla S > 0 \text{ and } S \in \mathcal{S} \,. \end{aligned} \tag{4}$$

Remark 1 *Notice how the objective in equation 4 does not contain the target density f making this construction amenable to cases when p is unknown or too complicated to evaluate.*

The optimization objective in equation 4 also corresponds to maximum likelihood-based estimation. Intuitively, the flow $S : \mathbb{R}^d \to \mathbb{R}^d$ maps a sample $\mathbf{x} \in \mathbb{R}^d$ to a latent variable $\mathbf{z} \in \mathbb{R}^d$ with tractable density g. This transformation implicitly defines the density of the original sample by the change of variable formula as

$$\tilde{p}(\mathbf{x}) := g(S(\mathbf{x}))|\det \nabla S(\mathbf{x})|\,.$$

Now S can be optimized by minimizing the negative log-likelihood of available samples furnishing the optimization objective in equation 4. Notice that calculating the $\det \nabla S(\mathbf{x})$ for any $\mathbf{x} \in \mathbb{R}^d$ is an $\mathcal{O}(d^3)$ operation. However, there is a myriad of machine learning architectures for defining the functions class $\mathcal{S}$ such that $\det \nabla S_i$ for each $i \in [k]$ can be performed with $\mathcal{O}(d)$ complexity Papamakarios et al. (2021).

In the context of domain adaptation, we want to find two transport maps f_S and f_T that transport $\mathcal{D}_S$ and $\mathcal{D}_S$ to Gaussian distribution respectively. Let $\{(\mathbf{x}_S^i, y_S^i)\}_{i=1}^n$ denote the labeled i.i.d. observations from the source domain and let $\{\mathbf{x}_T^i\}_{i=1}^n$ denote the unlabeled i.i.d. observations from the target domain. Then, the optimization problem we solve is

$$\hat{f}_S = \arg\min_{S\in\mathcal{S}} -\frac{1}{n}\sum_{i=1}^{n}[\log g \circ S(\mathbf{x}_S^i) + \log|\det \nabla S(\mathbf{x}_S^i)|] \quad \text{s.t.}\ \det \nabla S > 0 \text{ and } S \in \mathcal{S}\,, \tag{5}$$

and similarly,

$$\hat{f}_T = \arg\min_{S\in\mathcal{S}} -\frac{1}{n}\sum_{i=1}^{n}[\log g \circ S(\mathbf{x}_T^i) + \log|\det \nabla S(\mathbf{x}_T^i)|] \quad \text{s.t.}\ \det \nabla S > 0 \text{ and } S \in \mathcal{S}\,. \tag{6}$$

We use masked autoregressive flows Papamakarios et al. (2017) architecture to learn f_S and f_T.

4 Experiment: Amazon Review Dataset

We consider a simple setting of category-based source and target domain in the Amazon review dataset Ni et al. (2019). We use the Amazon dataset for distriobutionaly robust learning from the WILDS package Koh et al. (2021). One of the metadata fields in the WILDS Amazon dataset ascertains the category of product review from one among 27 categories. Among these, we choose reviews from the *Books* category as the source and reviews from *Home and Kithcen* as the target.

Out of the total $4,002,170$ data points in the Amazon dataset, there are a total of $2,779,625$ data points from the source category and a total of $126,421$ data points from the target category. While the label associated with each review gives a rating on a scale of 1 to 5, we collapse this 5-point rating to binary rating of *Positive* and *Negative* reviews where a negative review corresponds to ratings 1 to 3 and a positive review corresponds to ratings 4 and 5. We randomly choose a training set of size 5×10^3 and a test set of size 2.5×10^3 from both the source and target datasets. For these four subsets of data, the percentage of positive reviews is displayed in Table 1. For the feature representation of text reviews, we use BERT embeddings described in the next subsection.

	Source domain	Target domain
Train	58.56	63.52
Test	58.92	63.68

Table 1: Percentage of positive reviews in the train and test set of source and target dataset.

4.1 Finetune BERT for Embeddings

We use hidden layer embeddings of the text data as the feature representation of source and target domains. We use the embeddings from the last hidden layer of Google's BERT model for sequence

Metric	Source train split	Source test split	Target test split
Cross-entropy loss	0.4140	0.5705	0.6727
Accuracy	0.8150	0.7164	0.6612

Table 2: Performance of binary classifier on source and target splits.

classification. Before the embeddings are obtained for the source and target samples, we first fine-tune BERT for our classification task on a random subset of 2×10^4 labelled reviews.

Note that the subset of data chosen for fine-tuning does not necessarily belong to the two chosen source and target domains. The fine-tuning is done for two epochs of this subset data and takes around 10 minutes. The accuracy on the validation set at the end of two epochs of fine-tuning using BERT sequence classifier was 74.19%.

Finally, the BERT embeddings post fine-tuning are 768 dimensional. This is a very high dimensionality for binary classification and therefore, we use principal component analysis on normalized embeddings for dimension reduction. Figure 1 shows the scatterplot of the first two principal components of BERT embeddings for labeled source and target data. For the final classification task, we use first 16 principle components that explain 70% of variability in the data. For ease of notation, let us use the same notation for dimension-reduced BERT embeddings as the raw source and target data. That is, let $\{(\mathbf{x}_S^i, y_S^i)\}_{i=1}^{n_S}$ and $\{(\mathbf{x}_T^i)\}_{i=1}^{n_T}$ denote the labeled source and unlabeled target data respectively where each $\mathbf{x}_S^i, \mathbf{x}_T^j \in \mathbb{R}^{42}$ for $i \in [n_S]$ and $j \in [n_T]$.

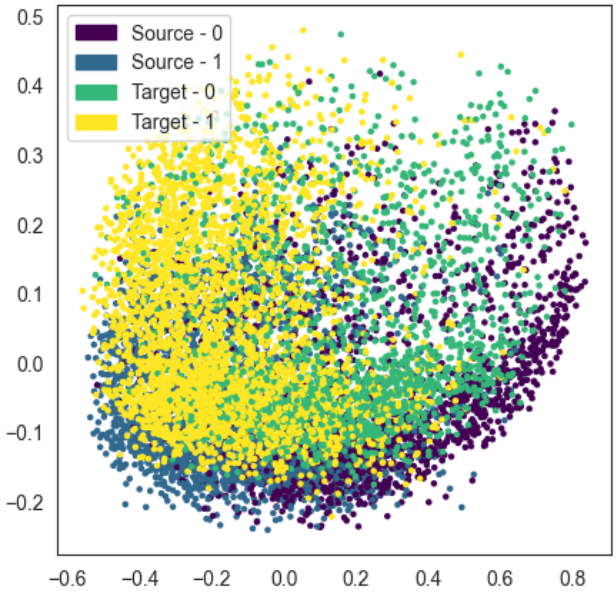

Figure 1: First two principle components of BERT embeddings with labels.

4.2 Training Binary Classifier on Source Domain

Before we create any transfer learning setup, we train a binary classifier on the labeled source data and study the gap between the classifier's performance on source and target data. We train a simple classifier with two linear layers and ReLU activation. Table 2 shows the cross-entropy loss and overall accuracy of the classifier on source and target split. Notice that on the test split of 2.5×10^3 samples, the gap in accuracy of source vs target domain is 0.0552.

4.3 LEARNING NORMALIZING FLOWS

For domain adaptation, we want to learn transport maps f_S and f_T via normalizing flows, specifically using masked autoregressive flow architecture Papamakarios et al. (2017). For convenience, we create a train and test split of source and target data of equal sizes. That is, we have 5×10^3 train samples from source and target distributions and use unsupervised learning problems equation 5 and equation 6 to estimate f_S and f_T respectively.

Figure 2 plots the quantile-quantile plots of the normalized source and target test data using their respective learned normalizing flows for a random set of five components. Notice the presence of outliers in the normalized data.

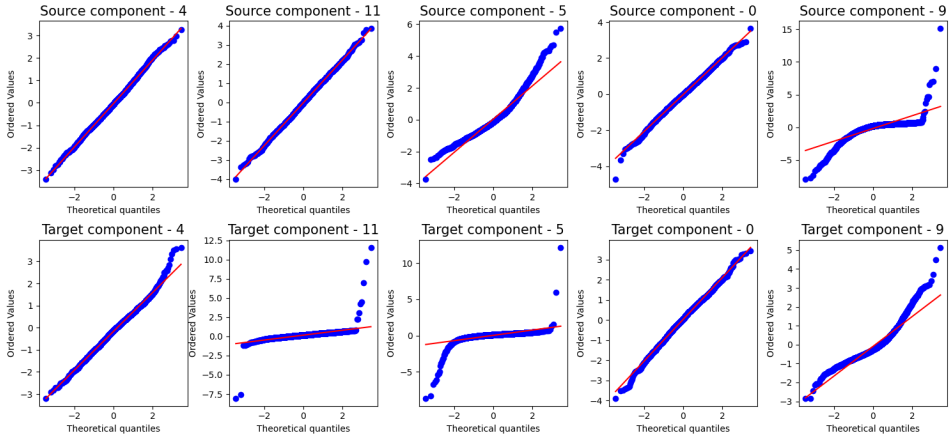

Figure 2: Gaussian quantile-quantile plots for 5 randomly components (out of 16) of the normalized source (top) and target (bottom) test split.

4.4 DOMAIN ADAPTATION RESULTS

Using the learned transport maps $\hat{f}_S$ and $\hat{f}_T$, we transport the test split of the target data to the source domain by applying the deterministic function $\hat{f}_S^{-1} \circ \hat{f}_T$ on the target samples. Using the binary classifier learned earlier, Table 3 shows the binary cross-entropy loss and accuracy of the classifier.

Metric	Source train split	Source test split	Target test split (UDA)
Cross-entropy loss	0.4140	0.5705	0.6960
Accuracy	0.8150	0.7164	0.5712

Table 3: Performance of binary classifier on source and target splits.

5 DISCUSSION

In this analysis, it becomes evident that the domain adaptation technique employed did not yield an enhancement in the accuracy of label predictions on the test subset of the target domain. This lack of improvement can likely be attributed to the presence of outliers within the normalized data, indicating potential errors in the estimation of transport maps. To address this limitation, we suggest exploring alternative normalizing flow architectures, as proposed by Kobyzev et al. (2020), for transporting BERT embeddings to a standard Gaussian distribution. The readers are advised to caution as the experiments conducted in this study were not finely tuned for optimal hyperparameters and did not fully explore the latest advancements in normalizing flows. Further exploration in this direction holds promise for enhancing the classifier's accuracy on target domain data.

REFERENCES

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

Minmin Chen, Kilian Q Weinberger, and Yixin Chen. Automatic feature decomposition for single view co-training. In *ICML*, volume 2, pp. 5, 2011.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *Advances in neural information processing systems*, 30, 2017.

Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43 (11):3964–3979, 2020.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. *Advances in neural information processing systems*, 21, 2008.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE transactions on neural networks*, 22(2):199–210, 2010.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

George Papamakarios, Eric T Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in neural information processing systems*, 20, 2007.

Kai Zhang, Vincent Zheng, Qiaojun Wang, James Kwok, Qiang Yang, and Ivan Marsic. Covariate shift in hilbert space: A solution via sorrogate kernels. In *International Conference on Machine Learning*, pp. 388–395. PMLR, 2013.