6.867 MACHINE LEARNING PROJECT STRATEGY DOCUMENT

Maohao Shen, Charles Lyu, Zhuo Liu

1 INTRODUCTION

The usual supervised learning for classification usually can achieve state-of-the-art results on common benchmarks, but may fail to generalize well when the models are deployed in real world. This is due to the fact that the training data and testing data are assumed to be sampled from the same distribution in the typical supervised learning problem. However, in a real world problem, we usually encounter the problem of the domain shift, where the distribution of the training data (referred as source domain) differs from the distribution of the data we want to test our model on (referred as target domain). Furthermore, obtaining the vast quantities of labeled data for the target domain for training is usually not practical. To address this problem, the unsupervised domain adaption aims to learn specific knowledge jointly from labeled source domain and unlabeled target domain, and produce a model with good generalization performance on both domains. One popular approach to achieve such goal is using the domain alignment. The key idea of domain alignment is to learn a feature extractor to extract common features in two domains, which lets the distribution of features across the target and source domains align together in the latent feature space.

However, since we cannot access the labels on target domain, simply aligning the source and target features cannot guarantee that the corresponding class conditional distribution will also be aligned correctly across the two domains. For example, in binary classification, the situation that negative samples in target domains wrongly aligning with positive samples in source domain can still lead to poor performance on target domain. Thus, the most important problem in unsupervised domain adaptation is that we should not only align the source and target features, but also align them correctly. To alleviate this problem, the Co-regularized Domain Alignment (Co-DA) algorithm proposed by Kumar et al. (2018) trains two different models for the source domain data and aligning the target domain feature distribution to each of them individually. An extra penalty is added when the predictions for two models disagree with each other, which helps in reducing the probability of "mis-alignment".

In this project, we will mainly study and analyze this method. Beyond that, we will also modify some parts of the algorithm and even incorporate new ideas to see if there will be improvements.

2 NOTATIONS AND PROBLEM FORMULATIONS

Suppose we have a labeled source domain dataset $\mathcal{D}_s := \{(\boldsymbol{x}_i^s, \boldsymbol{y}_i^s)\}_{i=1}^{n_s}$ with n_s training data, and an unlabled target domain dataset $\mathcal{D}_t := \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$, where we denote $\boldsymbol{x} \in \mathcal{X}$ as input data and $\boldsymbol{y} \in \mathcal{Y}$ as label. Source domain data $\boldsymbol{x}^s \sim P_s(\boldsymbol{x})$ and target domain data $\boldsymbol{x}^t \sim P_t(\boldsymbol{x})$ are sampled from different distributions, but the the underlying label $\boldsymbol{y}^t \in \mathcal{Y}$ shares the same label space as source domains. Our goal is to train two models $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ using both source domain and target domain data such that they can achieve good generalization performance on the unlabeled target domain. Each model $\boldsymbol{\theta}_j$ can be decomposed into a feature extractor $\boldsymbol{f}_j : \mathcal{X} \to \mathbb{R}^l$, followed by a classifier $\boldsymbol{h}_j : \mathbb{R}^l \to \mathbb{R}^{n_c}$, where l is the dimension of extracted feature embedding from model $\boldsymbol{\theta}_j$, and n_c is the number of classes. Thus, the prediction of input $\boldsymbol{x}$ using model $\boldsymbol{\theta}_j$ can be expressed as $\hat{\boldsymbol{y}}_j = \boldsymbol{\theta}_j(\boldsymbol{x}) = (\boldsymbol{h}_j \circ \boldsymbol{f}_j)(\boldsymbol{x})$. Let $\boldsymbol{f}_j \# P_s = \{\boldsymbol{f}_j(\boldsymbol{x}^s) : \boldsymbol{x}^s \sim P_s\}$ be the push-forward distribution of P_s in the latent feature space, and similarly define $\boldsymbol{f}_j \# P_t$.

3 METHODS

We first briefly describe the original Co-DA algorithm proposed by Kumar et al. (2018) in Section 3.1. We then evaluation some of their design choices in Section 3.2, including positives that we keep

as assumptions for our work, as well as areas of improvement that gives rise to our own designs in Section 4.

3.1 Description of original algorithm

Co-DA aims to find two models $\boldsymbol{\theta}_1 = \boldsymbol{h}_1 \circ \boldsymbol{f}_1$ and $\boldsymbol{\theta}_2 = \boldsymbol{h}_2 \circ \boldsymbol{f}_2$ that satisfy the following properties, which are expressed as losses in the objective function (1) below.

- **Accuracy**. Each $\boldsymbol{\theta}_j$ should classify the source samples $\mathcal{D}_s$ correctly, expressed as the cross-entropy loss $L_y(\boldsymbol{\theta}_i; P_s)$.
- **Domain alignment**. For each j, the disagreement between the alignment of feature distributions $\boldsymbol{f}_j \# P_s$ and $\boldsymbol{f}_j \# P_t$ should be minimized, measured by the Jensen-Shannon (JS) divergence as $L_d(\boldsymbol{f}_i \# P_s, \boldsymbol{f}_i \# P_t)$.
- **Target prediction alignment**. $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ should agree on predictions of target samples, that is, $\hat{\boldsymbol{y}}_1 = \hat{\boldsymbol{y}}_2$ for any $\boldsymbol{x} \in \mathcal{D}_t$. Their disagreement is captured by $L_p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; P_t)$.
- **Diverse feature extractors**. $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ should give different feature distributions for the *source* domain. Their diversity is expressed as a score $D_{\boldsymbol{f}}(\boldsymbol{f}_1, \boldsymbol{f}_2)$ to be maximized.

Kumar et al. (2018) use the following objective function to capture the above goals:

$$\min_{\substack{\boldsymbol{f}_i \in \mathcal{F}_i, \boldsymbol{h}_i \in \mathcal{H}_i \\ \boldsymbol{\theta}_i = \boldsymbol{h}_i \circ \boldsymbol{f}_i}} \mathcal{L}(\boldsymbol{\theta}_1) + \mathcal{L}(\boldsymbol{\theta}_2) + \lambda_p L_p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; P_t) - \lambda_{div} D_{\boldsymbol{f}}(\boldsymbol{f}_1, \boldsymbol{f}_2), \text{ where} \tag{1}$$

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}_i) &:= L_y(\boldsymbol{\theta}_i; P_s) + \lambda_d L_d(\boldsymbol{f}_i \# P_s, \boldsymbol{f}_i \# P_t) + \lambda_{sv} L_{vt}(\boldsymbol{\theta}_i; P_s) \\ &\quad + \lambda_{ce} \left(L_{ce}(\boldsymbol{\theta}_i; P_t) + L_{vt}(\boldsymbol{\theta}_i; P_t) \right), \end{aligned} \tag{2}$$

$$L_y(\boldsymbol{\theta}_i; P_s) := \mathbb{E}_{\boldsymbol{x}, \boldsymbol{y} \sim P_s} \left[\boldsymbol{y}^T \ln \boldsymbol{\theta}_i(\boldsymbol{x}) \right],$$

$$L_d(\boldsymbol{f}_i \# P_s, \boldsymbol{f}_i \# P_t) := \sup_{d_i} \mathbb{E}_{\boldsymbol{x} \sim P_s} \ln d_i(\boldsymbol{f}_i(\boldsymbol{x})) + \mathbb{E}_{\boldsymbol{x} \sim P_t} \ln(1 - d_i(\boldsymbol{f}_i(\boldsymbol{x}))), \tag{3}$$

$$L_p(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2; P_t) := \mathbb{E}_{\boldsymbol{x} \sim P_t} \left\| \boldsymbol{\theta}_1(\boldsymbol{x}) - \boldsymbol{\theta}_2(\boldsymbol{x}) \right\|_1,$$

$$D_{\boldsymbol{f}}(\boldsymbol{f}_1, \boldsymbol{f}_2) := \min \left(\nu, \left\| \frac{1}{b} \sum_{j=1, \boldsymbol{x}_j \sim P_s}^{b} (\boldsymbol{f}_1(\boldsymbol{x}_j) - \boldsymbol{f}_2(\boldsymbol{x}_j)) \right\|_2^2 \right), \tag{4}$$

where d_i is the *domain discriminator* that gives the probability that a given point in the feature space belongs to the source domain (implemented using a two-layer neural network), b is the batch size, ν is a hyperparameter limiting the maximum divergence, and $\lambda_p, \lambda_{div}, \lambda_d, \lambda_{sv}, \lambda_{ce}$ are regularization parameters that control the relative importance of each objective.

In addition, they also minimize the conditional entropy of $\boldsymbol{\theta}_i(\boldsymbol{x})$ and use virtual adversarial training (VAT), in order to encourage classification boundaries to be at low density regions:

$$L_{ce}(\boldsymbol{\theta}_i; P_t) = -\mathbb{E}_{\boldsymbol{x} \sim P_t} \left[\boldsymbol{\theta}_i(\boldsymbol{x})^T \ln \boldsymbol{\theta}_i(\boldsymbol{x}) \right], \tag{5}$$

$$L_{vt}(\boldsymbol{\theta}_i; P_t) = \mathbb{E}_{\boldsymbol{x} \sim P_t} \left[\max_{\|\boldsymbol{r}\| \leq \epsilon} D_{kl} \left(\boldsymbol{\theta}_i(\boldsymbol{x}) \| \boldsymbol{\theta}_i(\boldsymbol{x} + \boldsymbol{r}) \right) \right]. \tag{6}$$

3.2 Evaluation of original algorithm

We identified several advantages of Kumar et al. (2018)'s approach, which we will use as assumptions for our own work. On a high level, the objective (1) successfully captures all four goals of co-regularization. The choice of several loss functions, particularly the cross-entropy loss on source data L_y and the ℓ_1 distance for prediction disagreement L_p, are also standard well-established choices, so we maintain these design decisions. Notably, Kumar et al. (2018) justified their choice for the latter by noting that encouraging $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ to agree on predictions of P_t would suggest their alignments of feature distributions are likely similar with respect to classifier boundaries.

On the other hand, some aspects of Co-DA have alternative choices or can be improved. For example, Co-DA is based upon the state-of-the-art Virtual Adversarial Domain Adaptation (VADA) method

proposed by Shu et al. (2018), and inherits several components such as the domain discriminator d_i in (3) and L_{ce}, L_{vt} in (5) and (6). Such a design, while reasonable, makes the approach unsuitable for generic neural networks. Therefore, we will explore restructuring the model based on a typical convolutional neural network, with alternative measures of domain misalignment L_d, and compare them to the VADA-based approach as a baseline. This approach is detailed in Section 4.1.

It should be noted that while Co-DA had good empirical performance, it is still largely based on intuition with little theoretical guarantees. However, we will not be conducting theoretical analysis on the model due to time constraints.

4 Analysis and Potential Improvement

We plan to study this unsupervised domain adaptation method from two main perspectives: model structure used for domain adaptation, as well as designing new algorithms beyond this method and further improve the performance.

4.1 Model Structure

As shown in equation 3, Co-DA uses a discriminator to enforce domain alignment between source and target domain. However, such an adversarial training approach cannot apply to general models, such as the typical neural network without a domain discriminator $\boldsymbol{d}_j$. Thus, we will try to use an alternative typical neural network as our domain adaptation model. However, without the discriminator to measure the distance between two distributions, we need use other techniques to enforce domain alignment. For instance, we can use Maximum Mean Distribution(MMD) loss (Gretton et al., 2012) to measure the distribution distance, i.e.:

$$\begin{aligned}\mathcal{L}_{\text{MMD}} =& \frac{1}{N_s(N_s-1)}\sum_{i=1}^{N_s}\sum_{i'=1}^{N_s}\mathcal{K}\left(\boldsymbol{f}_j(\boldsymbol{x}_i^s),\boldsymbol{f}_j(\boldsymbol{x}_{i'}^s)\right)+\frac{1}{N_t(N_t-1)}\sum_{i=1}^{N_t}\sum_{i'=1}^{N_t}\mathcal{K}\left(\boldsymbol{f}_j(\boldsymbol{x}_i^t),\boldsymbol{f}_j(\boldsymbol{x}_{i'}^t)\right)\\ &-\frac{2}{N_sN_t}\sum_{i=1}^{N_s}\sum_{i'=1}^{N_t}\mathcal{K}\left(\boldsymbol{f}_j(\boldsymbol{x}_i^s),\boldsymbol{f}_j(\boldsymbol{x}_{i'}^t)\right)\end{aligned}$$

Where $\mathcal{K}(\boldsymbol{x},\boldsymbol{y})$ denotes the kernel function, for instance, the Gaussian kernel: $\mathcal{K}(\boldsymbol{x},\boldsymbol{y}) = \exp\left(-\frac{1}{2\sigma}\left\|\boldsymbol{x}-\boldsymbol{y}\right\|^2\right)$, and N_s and N_t denotes the number of source data and target data in a batch to compute the MMD loss, respectively.

4.2 Potential Novel Ideas and Improvements

Most existing unsupervised domain adaptation methods mainly focus on domain alignment in feature space, which only use the supervision in labeled source domain. However, viewing the problem from another perspective, we can actually use self-training techniques to help improve the performance of model predictions on target domain. The simple way to do this is using pseudo labeling: as we have two models $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ (or more generally m models), we can generate good-quality pseudo labels on a subset of the target dataset based on both the confidence and agreement of the models. To quantify this, we first compute a score on each target data:

$$s(\boldsymbol{x}^t)=\frac{1}{m}\sum_{j=1}^{m}\mathcal{H}\left(P\left(\boldsymbol{y}|\boldsymbol{x}^t;\boldsymbol{\theta}_j\right)\right)+\mathcal{H}\left(\frac{1}{m}\sum_{j=1}^{m}P\left(\boldsymbol{y}|\boldsymbol{x}^t;\boldsymbol{\theta}_j\right)\right)$$

Where $\mathcal{H}$ denotes the entropy function. Intuitively, a small $s(\boldsymbol{x}^t)$ means each individual model $\boldsymbol{\theta}_j$ is confident about its prediction, and they also agree with each other with high confidence, which implies that their prediction on target data is almost correct and can be used to generate pseudo label. Thus, the pseudo label of each target data can be defined as:

$$\tilde{\boldsymbol{y}}=\begin{cases}\arg\max_{\boldsymbol{y}}\frac{1}{m}\sum_{j=1}^{m}P(\boldsymbol{y}|\boldsymbol{x}^t;\boldsymbol{\theta}_j) & \text{if } s(\boldsymbol{x}^t)<\lambda\\ \emptyset & \text{Otherwise}\end{cases}$$

Where λ is a threshold as a hyper-parameter. A pseudo label is only assigned to a data point if the models are confident about its predictions, and we only use target data with pseudo labels for training.

5 Empirical Implementation

Similar as Kumar et al. (2018), we are going to treat the results from Virtual Adversarial Domain Adaptation (VADA) method proposed by Shu et al. (2018) as a baseline. The first step will be reproducing the improvement from VADA by implementing the Co-DA. After that, we plan to implement our new ideas as described in the previous section and compare them with the results from VADA and original Co-DA.

The Co-DA has two predictors, where each predictor is described by VADA. The model architecture of VADA has a feature generator $\boldsymbol{f}$, a feature classifier $\boldsymbol{h}$, and a domain discriminator d. The data classifier $\boldsymbol{\theta} = \boldsymbol{h} \circ \boldsymbol{f}$ consists of convolution, max-pool, dropout and Gaussian noise layers with eighteen layers in total. The domain discriminator d is a neural network with two fully connected layer, which takes the output from $\boldsymbol{f}$ as input and outputs the probability of the input sample belonging to the source domain. The objective of VADA model for learning a data classifier $\boldsymbol{\theta} = \boldsymbol{h} \circ \boldsymbol{f}$ is given in (2) as $\mathcal{L}(\boldsymbol{\theta}_i)$. For Co-DA, the two VADA model has the same architecture but initialized with different random seeds, and the overall objective is given in (1). We are going to use the hyper-parameters described in Kumar et al. (2018).

The datasets we are going to use to evaluate the results are those different digits dataset including MNIST (LeCun et al., 1998) and SVHN (Netzer et al., 2011). Both MNIST and SVHN are digit images but differ greatly in style. MNIST consists of gray-scale handwritten digits whereas SVHN consists of house numbers from street view images. Either one can be treated as the source domain and the other will be the target domain.

6 Future Work

If we still have time, we may continue exploring and analyzing this algorithm by changing the number of models used for training, as well as analyzing the trade-off between different objectives.

6.1 Multi-regularized Domain Adaptation

As long as there's enough GPU RAM, it is also interesting to try training more than two models (multi-regularized instead of co-regularized). Intuitively, more models can further mitigate the issue of wrong alignment between source domain and target domains with respect to labels. Assume we have trained multiple models $\boldsymbol{\theta}_j$ and they are independent with each other. Given a input target data $\boldsymbol{x}^t$ with unknown ground-truth label $\boldsymbol{y}^t$, each model will output a prediction, denote it as $P(\boldsymbol{y}|\boldsymbol{x}^t; \boldsymbol{\theta}_j)$. Because we enforce the multiple models agree with each other, we can derive the probability that the prediction of data $\boldsymbol{x}^t$ is correct given the model agree with each other:

$$P(\boldsymbol{y} = \boldsymbol{y}^t) = \frac{\prod_{j=1}^{m} P(\boldsymbol{y} = \boldsymbol{y}^t|\boldsymbol{x}^t; \boldsymbol{\theta}_j)}{\sum_{c=1}^{n_c} \prod_{j=1}^{m} P(\boldsymbol{y} = c|\boldsymbol{x}^t; \boldsymbol{\theta}_j)}$$

Where c denotes different classes, and m denotes the number of models.

Proposition 6.1. *If each model's prediction has confidence more than one-half, i.e.,* $P(\boldsymbol{y}|\boldsymbol{x}^t; \boldsymbol{\theta}_j) > 0.5$*, and the number of model goes to infinity, i.e.,* $m \to \infty$*, we can show that the prediction of target data* $\boldsymbol{x}^t$ *with all model agreement has to be correct, i.e.:*

$$P(\boldsymbol{y} = \boldsymbol{y}^t) = \lim_{m \to \infty} \frac{\prod_{j=1}^{m} P(\boldsymbol{y} = \boldsymbol{y}^t|\boldsymbol{x}^t; \boldsymbol{\theta}_j)}{\sum_{c=1}^{n_c} \prod_{j=1}^{m} P(\boldsymbol{y} = c|\boldsymbol{x}^t; \boldsymbol{\theta}_j)} = 1$$

6.2 Trade-off different objectives

It's also interesting to study the contribution of different objectives to the performance. For example, the diverse feature loss shown in equation 4 and the number of models as discussed in Section 6.1 should be closely relevant. Does there exist a trade-off between feature diversity and number of models? Some theoretical analysis on the relationship between different objectives is also interesting, although it might be beyond the scope of this project.

REFERENCES

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, William T Freeman, and Gregory Wornell. Co-regularized alignment for unsupervised domain adaptation. *arXiv preprint arXiv:1811.05443*, 2018.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Rui Shu, Hung H. Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation, 2018.