Importance Reweighting Using Adversarial-Collaborative Training

Yifan Wu
yw4@andrew.cmu.edu

Tianshu Ren
tren@andrew.cmu.edu

Lidan Mu
lmu@andrew.cmu.edu

Abstract

We consider the problem of reweighting a source dataset D_S to match a target dataset D_T, which plays an important role in dealing with the covariate shift problem. One of the common approaches to reweight the source data to match the distribution of target data is to use kernel mean matching, which tries to learn the likelihood ratios by minimizing the kernel mean discrepancy. In this work, we first drive a counterpart for the kernel mean matching technique by replacing the kernel mean discrepancy with the adversarial training objective. Then we argue that likelihood ratio based reweighting may not be the best choice for the covariate shift problem in terms of low effective sample size. To balance between the distribution matching and the effective sample size, we further propose another learning objective that contains a "collaborator" in addition to the adversary. The effectiveness of our approach is shown by preliminary experiment results.

1 Introduction

Covariate shift is a common problem when dealing with real world machine learning problems. It is quite often that the training data and test data come from different distributions. Existing approaches [1, 2, 3, 4] often contain two stages: The first stage is a "data reweighting" process, where the training data are reweighted such that its distribution is matched to the test data (in this stage only the feature vectors are considered, without any labels). And the second stage is to train the model with "reweighted" loss functions. Here we only consider the first stage.

One way of reweighting the data is called kernel mean matching [2], where the weights over the training data are optimized to minimize the kernel mean discrepancy. In kernel meaning matching, the kernel mean discrepancy can be viewed as a divergence measure of two sample sets. [5] proposed another way to measure the divergence between two sample sets by introducing an "adversarial discriminator". The intuition is that the two sample sets are more likely to be from the similar distributions if they are harder to be differentiated by a classifier.

In this work, we first investigate the possibility of replacing the kernel mean matching objective with adversarial training. We empirically show that, on simple synthetic datasets, the importance weights can be learned by the adversarial training process. Then we argue that, in covariate shift [1] or other potential applications, importance reweighting based on matching the likelihood ratios may not be the most effective choice due to the possibly low effective sample size. So we further propose an approach that is able to balance between distribution matching and effective sample size by adding a "collaborator". The idea is that, first we can interpret the importance weights as the acceptance rate in doing subsampling from the source data. Then we can push up the effective sample size by simultaneously maximizing the discrepancy between the rejected samples and the target dataset. In

[1]here we only consider the reweighting technique, excluding the discussion of transfer learning in domain adaptation as in [6]

other words, we will be able to sample as much relevant data as possible from the source dataset in order to perform further tasks with the target dataset.

The learning outcome of our adversarial-collaborative training can be also viewed as a classifier between population A and B. The classifier is learned from a sample set of population A and a mixed unlabelled sample set $A \cup B$, which is a semisupervised classification problem with labelled data only from one class and unlabelled data. To the best of our knowledge, we are not aware of any algorithm that can directly learn the classifier in this setting. This learning outcome may also have other potential applications when attention schemes are needed.

2 Problem Statement

Suppose we have a source data distribution $p_S(\cdot)$, a target data distribution $p_T(\cdot)$, and we are doing rejection sampling from $p_S(\cdot)$ to match $p_T(\cdot)$ with acceptance probability $\beta(\cdot) \in [0, 1]$. We define the distribution of the sample outcomes as $p_{\beta(S)}(\cdot)$ where $p_{\beta(S)}(x) = \frac{p_S(x)\beta(x)}{\int \beta(x) p_S(x) dx}$. The goal is to learn $\beta(\cdot)$ from finite samples $D_S \sim p_S(\cdot)$ and $D_T \sim p_T(\cdot)$ to minimize $\mathbb{D}\left(p_{\beta(S)}, p_T\right)$ for some divergence measure $\mathbb{D}(p, q)$. It can be shown that this problem is the equivalent to kernel mean matching [2] when $\mathbb{D}(p, q) = \|\mathbb{E}_{x\sim p}[\Phi(x)] - \mathbb{E}_{x\sim q}[\Phi(x)]\|_{\mathcal{H}}$ for some feature map $\Phi(\cdot)$ implicitly defined by a kernel function. The optimal solution is that β is proportional to the likelihood ratio, i.e.

$$\beta(x) \propto \frac{p_T(x)}{p_S(x)}$$

for all x, such that $p_{\beta(S)}$ and p_T are the same distribution.

Note that here we interpret β as "acceptance probability" in rejection sampling instead of "importance reweighting" because it brings convenience to define a distribution $p_{\beta(S)}(\cdot)$ over x and use the term "divergence measure". Since under both interpretation the goal is just to learn the likelihood ratio we can use learned β as either acceptance probabilities or importance weights.

Different from kernel mean matching where an individual weight β_x is learned for all data point $x \in D_S$, our goal here is to learn a function mapping $\beta(\cdot)$ that maps the feature space to $[0, 1]$, which provides (i) smoothing on the weights such that similar points will have similar weights and (ii) the ability of generalization to unseen data and may have potential more applications other than the simple covariate shift setting.

3 Algorithms

In this section we introduce two approaches to learn $\beta(\cdot)$. The first one is trying to learn the likelihood ratio as in kernel mean matching. After discussing some potential issues with the first approach, we introduce another algorithm by adding a collaborative classifier, which pushes $\beta(\cdot)$ to be a more "sample efficient" importance reweighter.

3.1 Importance reweighting with adversarial training

In the work of Generative Adversarial Nets [5], the goal is to learn a generative model that indirectly models the data distribution as a multi-layer perceptron by using finite data samples. It learns the generative model f by introducing an adversary g (a binary classifier) that tries to differentiate between samples generated from f and samples from the true data distribution p_{data}. Specifically, g is trained to make the best classification while f is trained to minimize the power of g. There training objective can be written as $\min_f \mathbb{D}(f, p_{data})$ where $\mathbb{D}(p, q) = \max_g \mathbb{E}_{x\sim p}[\log g(x)] + \mathbb{E}_{x\sim q}[\log(1 - g(x))]$.

Here our goal is to use this divergence measure to learn $\beta(\cdot)$, that is

$$\min_{\beta} \max_{g} \mathbb{E}_{x\sim p_{\beta(S)}}[\log g(x)] + \mathbb{E}_{x\sim p_T}[\log(1 - g(x))] \,. \tag{1}$$

To learn from finite number of samples for both source and target datasets, we derive the empirical optimization formulation for (1): Define

$$f_1(\theta, \lambda) = \frac{1}{\sum_{x\in D_S} \beta_\theta(x)} \sum_{x\in D_S} \beta_\theta(x) \log g_\lambda(x) + \frac{1}{|D_T|} \sum_{x\in D_T} \log(1 - g_\lambda(x)) \,,$$

and the goal is to optimize

$$\min_{\theta} \max_{\lambda} f_1(\theta, \lambda)\,. \tag{2}$$

Optimizing (2) can be done by performing gradient descent/ascent on θ and λ alternatively [2]. Note that, for large datasets, optimizing over λ can be done by stochastic gradient descent but optimizing θ cannot. This is because the gradient has $\sum_{x\in D_S}\beta(x)$ in the denominator. So we cannot use stochastic gradient descent but a mini-batch optimization instead to get a good estimation of $\sum_{x\in D_S}\beta(x)$.

One problem with this algorithm is that $\beta_\theta(x)$ can be arbitrarily small (as long as it is proportional to the likelihood ratio) if we optimize the above objective (2). It is fine to use the small weights for sample reweighting in the covariate shift scenario but if the goal of learning β is to do rejection sampling (as described in Section 2) with a finite number of available source samples D_S then small $\beta_\theta(x)$ will lead to low acceptance rate, which means that we ignore a lot of useful samples. So attempting to accept as many samples that are likely from the target distribution as possible is one of the motivations that we propose the second algorithm.

Another motivation is a general issue with sample reweighting in the covariate shift scenario: the bias variance trade-off (as discussed in [3]). Using the likelihood ratio as importance weights will give an unbiased estimate of the Bayes risk on the test data distribution. However, the importance weights may lead to high variance due to the possibly low effective sample size $\frac{\|\beta\|_1^2}{\|\beta\|_2^2}$. Moreover, in learning tasks without model misspecification, it can be shown that, even if there is a mismatch between the training and test distributions, doing empirical risk minimization **without** any reweighting is the optimal thing to do as discussed in [7]. Since assuming no model misspecification is kind of too strong, a good reweighting should balance between the distribution matching (no bias) and the highest effective sample size (no reweighting). Here what we are trying to achieve is to push $\beta(x)$ to be 1 for all x that is a point of interest according to the target dataset while pushing $\beta(x)$ to be 0 if x is not a point of interest, which will lead to a balance between the distribution matching and high effective sample size. [3]

3.2 Sample-efficient reweighting with adversarial-collaborative training

Motivated by the issues discussed above, here we propose an algorithm that learn a $\beta(\cdot)$ that can both (i) sample as many points of interest in the source dataset as possible (have high effective sample size) (ii) reject points that are not likely from the target distribution. The idea is to simultaneously maximize the divergence between the rejected samples and the target samples using a "collaborator" h. That is, by defining

$$f_2(\theta,\lambda) = \frac{1}{\sum_{x\in D_S} 1-\beta_\theta(x)} \sum_{x\in D_S} (1-\beta_\theta(x)) \log h_\lambda(x) + \frac{1}{|D_T|}\sum_{x\in D_T} \log(1-h_\lambda(x))\,,$$

we want to optimize

$$\min_{\theta} \left(\max_{\lambda_1} f_1(\theta,\lambda_1) - \gamma \max_{\lambda_2} f_2(\theta,\lambda_2) \right)\,. \tag{3}$$

The parameter $\gamma > 0$ controls the balance between distribution matching and effective sample size. Small γ will make learned β closer to the likelihood ratio while large γ will make learned β have higher effective sample size. Note that when $\gamma = 0$ the objective is the same as (2).

4 Experiments

In this section, we empirically show that our proposed algorithms can achieve certain properties.

[2]Here we will not discuss which optimization technique is good for optimizing the objective since how to optimize the adversarial training objective with good convergence guarantee is still an open problem.

[3]We are not arguing that this will lead to the optimal balance.

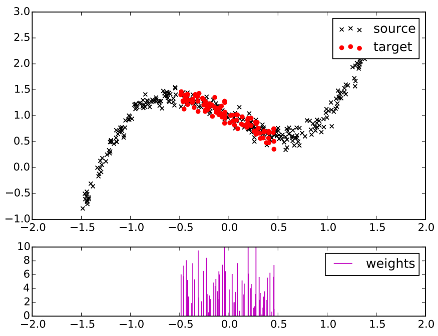
4.1 Synthetic dataset

We generate a one-dimensional synthetic dataset by sampling source dataset $D_S \sim \text{Uniform}(-1.5, 1.5)$ and target dataset $D_T \sim \text{Uniform}(-0.5, 0.5)$. We further generate labels $y = x^3 - x + 1 + \epsilon$, where $\epsilon \sim N(0, 0.1^2)$ to show why we may need reweighting to do regression (e.g. if we use linear regression here it is better to use data only from $[-0.5, 0.5]$). In our experiment, the number of source samples is 300 and the number of target samples is 100.

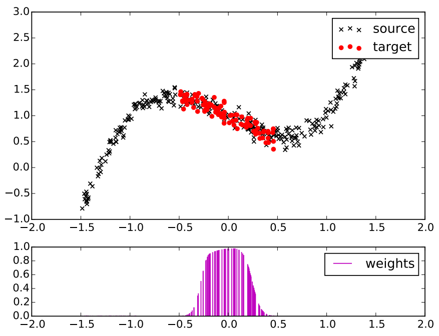
4.1.1 Visualizing learned weights

We compare the learned weights from our algorithms with kernel mean matching, where we use a rbf kernel with radius 1. In our algorithms all β, g and h are $1 \rightarrow 16 \rightarrow 6 \rightarrow 1$ fully connected neural networks with relu activation functions. In each training step we train g and h for one epoch using SGD and train β for 10 epochs using mini-batch with batch-size 30.

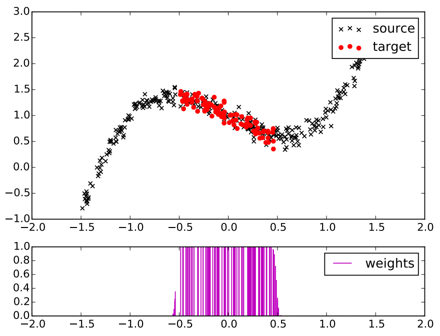
Figure. 1 shows the learned β by different algorithms. We can see that the weights learned by either kernel mean matching or the naive adversarial training in section 3.1 suffer from low effective sample size. But the adversarial-collaborative training algorithm we proposed do have expected properties (reweighting all relevant data samples uniformly while reject irrelevant ones). More interestingly, since we generate points from uniform distributions the likelihood ratios in $[-0.5, 0.5]$ should be same, which means that our second algorithm performs even better on learning β to be the likelihood ratio although its objective is sort of different.



(a) Kernel mean matching



(b) Adversarial training ($\gamma = 0$)



(c) Adversarial-collaborative training ($\gamma = 1$)

Figure 1: Visualizing learned β on 1-D synthetic dataset

4.1.2 Convergence of the learning objective

Figure 2 shows the convergence of the crossentropy loss for the adversary g_{λ_1} and the collaborator h_{λ_2} ($-f_1$ and $-f_2$ respectively), from which we can see that, as expected, the loss of g_{λ_1} converges to a higher level while the loss of h_{λ_2} converges to a lower level.

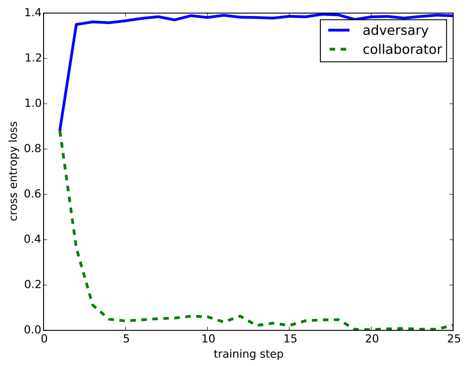


Figure 2: Convergence of the loss for adversary g and collaborator h.

4.2 MNIST dataset

We also experiment on the MNIST dataset where we randomly sample 3000 images as the source dataset. For target dataset, we sample 300 images from the set of digit 0. Table 1 shows the proportions that each digit is sampled from the source set according to learned $\beta(\cdot)$ (accept $\beta(x) > 1/2$) by our adversarial-collaborative algorithm.

digits	0	1	2	3	4	5	6	7	8	9
acceptance rate	0.987	0	0.057	0.037	0.013	0.08	0.06	0.043	0.02	0.03

Table 1: Performance on MNIST dataset

Results show that our learned sampler can successfully sample 0 while rejecting other digits. This result is only based on the training source dataset D_S and we have not studied the generalization ability of the learned sampler.

One thing to be mentioned here is that in this task kernel mean matching with 2-degree-polynomial kernel can perform perfectly (select all 0 and reject all others). This might be because our model has many hyperparameters to tune and is hard to train while polynomial kernel is known to be powerful on the MNIST dataset. The value of our approach is that it is flexible and has the potential to handle very complex tasks (because any binary classifier can be used as β, g and h) and large datasets (can be trained with SGD and minibatch) where kernel mean matching might be too slow (due to large sample size) or not powerful enough (in terms of learning representations).

5 Conclusion and Future Work

In this work we propose an alternative approach to perform importance reweighting in the covariate shift scenario using adversarial training. We also propose an adversarial-collaborative training objective to learn importance weights that are balanced between the effective sample size and the distribution matching (bias). Experimental results show that our approach is able to achieve certain properties.

Future tasks include investigating (i) whether our approaches are applicable on more complicated real datasets, (ii) whether it actually helps the covariate shift tasks or other possible applications and (iii) theoretical analysis.

References

[1] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(Sep):2137–2155, 2009.

[2] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[3] Sashank J Reddi, Barnabas Poczos, and Alex Smola. Doubly robust covariate shift correction. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2949–2955. AAAI Press, 2015.

[4] Junfeng Wen, Russell Greiner, and Dale Schuurmans. Correcting covariate shift with the frank-wolfe algorithm. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1010–1016. AAAI Press, 2015.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[6] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1180–1189, 2015.

[7] Junfeng Wen, Chun-Nam Yu, and Russell Greiner. Robust learning under uncertain test distributions: Relating covariate shift to model misspecification. In *ICML*, pages 631–639, 2014.