Abstract

Distribution shift is a major obstacle to the deployment of current deep learning models on real-world problems. Let $Y$ be the target (label) and $X$ the predictors (features). We focus on one type of distribution shift, target shift, where the marginal distribution of the target variable $P_Y$ changes but the conditional distribution $P_{X|Y}$ does not. Existing methods estimate the density ratio between the source- and target-domain label distributions by density matching. However, these methods are either computationally infeasible for large-scale data or restricted to shift correction for discrete labels. In this paper, we propose an end-to-end Label Transformation Framework (LTF) for correcting target shift, which implicitly models the shift of $P_Y$ and the conditional distribution $P_{X|Y}$ using neural networks. Thanks to the flexibility of deep networks, our framework can handle continuous, discrete, and even multidimensional labels in a unified way and is scalable to large data. Moreover, for high dimensional $X$, such as images, we find that the redundant information in $X$ severely degrades the estimation accuracy. To remedy this issue, we propose to match the distribution implied by our generative model and the target-domain distribution in a low-dimensional feature space that discards information irrelevant to $Y$. Both theoretical and empirical studies demonstrate the superiority of our method over previous approaches.

1. Introduction

Standard supervised learning methods typically assume that the training set (source domain) and the test set (target domain) have the same distribution. However, the data available for training is always limited and may not represent and reflect the statistics of the test data. As such, the source-domain distribution $P^S_{X|Y}$ is often different from the target-domain distribution $P^T_{X|Y}$, degrading the performance of the models learned on the training set. This phenomenon is called distribution shift, which has become a major obstacle to the deployment of deep learning models in the real world.

To overcome distribution shift and improve the prediction on test data, existing methods have studied various distribution shift settings, among which covariate shift and target shift have been widely considered. Covariate shift assumes that the marginal $P_X$ changes across training and test sets, whereas the conditional distribution $P_{Y|X}$ is invariant (Shimodaira, 2000; Sugiyama et al., 2008; Gretton et al., 2009; Long et al., 2015; 2017; 2018; Liu et al., 2019). Target shift assumes that the label distribution $P_Y$ changes but the conditional distribution $P_{X|Y}$ stays the same (Zhang et al., 2013; Iyer et al., 2014; Lipton et al., 2018; Azizzadenesheli et al., 2019).

Here we focus on the target shift problem, since it appears in a wide range of real-world learning problems. For example, in disease prediction, where our goal is to predict disease $Y$ from symptoms $X$, the distribution of the disease can change over location and time, while the mechanism of symptoms $P_{X|Y}$ is rather stable. Consider the flu prediction task, the data available for flu prediction is always has a regular morbidity rate, but if a model is trained on these data, the performance of this model will decrease when it is used to detect flu in a location or over a period with a high morbidity rate (Tasche, 2017). In addition, target shift also exists in many computer vision applications, such as predicting object locations (Yang et al., 2018) ands direction and human poses (Martinez et al., 2017). The distribution of object locations or human poses often changes across training and test sets.

Despite being a natural phenomenon in many real applications, target shift is relatively understudied compared to covariate shift. Chan & Ng (2005) proposed an expectation–maximization algorithm that requires estimation of the...
conditional distribution $P_{X|Y}$. Unfortunately, estimating $P_{X|Y}$ is difficult for high-dimensional $X$ and moreover, it does not apply to regression problems. Zhang et al. (2013) proposed a nonparametric method to estimate the density ratio $P_X^S / P_X^T$ by kernel mean matching of distributions, which is applicable to both regression and classification problems. However, this approach is not compatible with large data as the computational cost is quadratic in the sample size. Recently, Lipton et al. (2018); Azizzadenesheli et al. (2019) proposed efficient and sample size-independent methods that make use of the confusion matrix of a classifier learned on the training set. These methods have shown promising performance on large-scale data but are only applicable to classification problems.

In this paper, we aim to propose a new framework that can correct target shift for both discrete and continuous $Y$. Compared to existing methods, we make the following contributions. First, instead of estimating the density ratio $P_X^T / P_X^S$, we model the change in the distribution $P_Y$ by a neural label transformation $T$, which transforms the training label distribution $P_Y^S$ to a new label distribution $P_Y^B$ that can approximate the unknown $P_Y^T$ in the test set. Thanks to the flexibility of neural nets, we can design different transform models $T$ to deal with different types of $Y$, including discrete, continuous, and even multi-dimensional labels. Second, because of the absence of labels in the test set, we model the invariant conditional distribution $P_{X|Y}^T$ using a conditional generator $G$ on the training set. By concatenating the label transformation model $T$ with the conditional generator $G$, we can generate corresponding sample distribution $P_X^B$, which is then matched with $P_X^T$ to estimate the parameters in $T$. Third, for high dimensional $X$, such as images, we observe that the redundant information significantly degrades the estimation accuracy. To remedy this issue, we theoretically analyze this phenomenon and propose to match the distributions of a feature representation of $X$ that discards the information irrelevant to $Y$.

To demonstrate the advantage of our framework in practical applications, we apply our method to a range of label types, including classification (discrete label), regression (continuous label) and objects 2D object position prediction (multi-dimension label), in various target shift settings, such as random target shift, high probability label quantification and low probability label quantification. The empirical results demonstrate the generality, flexibility and superiority of our framework compared to previous methods.

2. Related Work

Covariate shift and target shift are two common types of distribution shift. The former one assumes that the feature distribution $P_X$ changes over training set and test set, but the conditional distribution $P_{Y|X}$ from label to data remains unchanged, while the latter one assumes that the label distribution $P_Y$ changes but $P_{X|Y}$ is invariant.

The existing methods solve covariate shift and target shift using re-weighting methods, which are also used in a wide range of problems, e.g., label-noise (Liu & Tao, 2015; Yu et al., 2017b; Cheng et al., 2017; Fang et al., 2020). We first introduce methods dealing with covariate shift problems shortly, where many methods estimate importance sample weights $P_X^S / P_X^T$ (Zadrozny, 2004; Huang et al., 2007; Sugiyama et al., 2008; Gretton et al., 2009) via kernel methods (Huang et al., 2007; Gretton et al., 2009; Zhang et al., 2013) or using a discriminative classifier (Lopez-Paz & Oquab, 2016; Liu et al., 2017). Then they correct models by retraining a new model with re-weighted training samples using estimated $P_X^S / P_X^T$ under the ERM framework (Shimodaira, 2000). More recent works learn domain-invariant representations $X' = h(X)$ that have similar marginal distributions across domains ($P_{X'}^T \approx P_{X'}^S$) (Si et al., 2009; Pan et al., 2010; Baktashmotlagh et al., 2013; Tzeng et al., 2014; Ganin et al., 2016; Long et al., 2015).

Similar to the correction of Covariate shift, there are two major steps to solve target shift problems. The first step is to estimate the label distribution $P_Y^T$ in the target domain or the ratio $P_Y^T / P_Y^S$. The second step is to construct an unbiased estimate of the target domain risk based on the results from the first step. Zhang et al. (2013); Iyer et al. (2014); Nguyen et al. (2016); Gong et al. (2016) proposed to estimate $P_Y^T$ or $P_Y^T / P_Y^S$ by matching a weighted combination of conditionals $P_{X|Y}^S$ in the source domain the marginal distribution $P_X^T$ in the target domain. The matching of distributions is achieved by minimizing suitable divergence measures (Gretton et al., 2012; Sugiyama et al., 2012) w.r.t. the weights on $P_{X|Y}^S$. In the discrete $Y$ scenario, Lipton et al. (2018) proposed a method which estimates the importance weight ($P_Y^T / P_Y^S$) by matching the output of trained classifier on the training set (confusion matrix), and then Azizzadenesheli et al. (2019) turned this problem as a linear programming problem and iteratively minimized the error of label distributions between the training set and the test set, improving the accuracy of estimated target label distribution $P_Y^T$. In addition, Azizzadenesheli et al. (2019) added a regularization term to make the algorithm compatible with the situation where the target sample size is small.

3. Methodology

Given training data $D = \{x_i^s, y_i^s\}_{i=1}^{n_s} \subseteq X \times Y$ independently drawn from an unknown joint distribution $P_{X,Y}^S$, denoted as the source domain distribution, and test data $D_t = \{x_i^t, y_i^t\}_{i=1}^{n_t}$ drawn from the target-domain distribution $P_{X,Y}^T$, where $y_i^t$ is unknown, target shift assumes that $P_{X|Y}^S = P_{X|Y}^T = P_{X|Y}$ and $P_Y^T \neq P_Y^S$. Our goal is to build
a model to estimate the label distribution $P^T_Y$ in the target domain such that we can correct the label shift between the training and test data and thus improve the prediction performance on the test set. We consider both continuous $Y$, i.e., $Y = \mathbb{R}^d$, and discrete $Y$, i.e., $Y = \{1, \ldots, K\}$.

### 3.1. Review of Previous Methods

To estimate the label distribution $P^T_Y$, existing methods use the relation between source and target distributions:

$$P^T_X(x) = \int_y P_{X|Y}(x|y)P^T_Y(y)dy$$  \hspace{1cm}  (1)

$$= \int_y P^S_{X|Y}(x, y)P^T_Y(y)dy. \hspace{1cm}  (2)$$

Because $P^S_{X|Y}$ and $P^T_X$ can be estimated from $D_s$ and $D_t$, previous methods (Zhang et al., 2013; Gong et al., 2016) estimate the density ratio $\beta^*(y) = \frac{P^T_Y(y)}{P^S_Y(y)}$ by minimizing the empirical Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) between $P^T_X$ and $P^S_X$:

$$||\frac{1}{n_s} \sum_{i=1}^{n_s} \psi(x_i^s) - \frac{1}{n_t} \sum_{i=1}^{n_t} \beta(y_i^t)\psi(x_i^t)||^2_{\mathcal{H}}, \hspace{1cm}  (3)$$

$$s.t. \beta(y_i^t) \geq 0, \text{ and } \sum_{i=1}^{n_s} \beta(y_i^t) = n_s, \hspace{1cm}  (4)$$

where $\psi$ is the feature mapping from $\mathcal{X}$ to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ associated with a kernel function $k(x, x') = \langle \psi(x), \psi(x') \rangle_{\mathcal{H}}$. For kernel functions that have no explicit $\psi$, for example, RBF kernels, we need to use kernel trick to calculate (3). The computational cost is quadratic in the sample size and thus the algorithm is not scalable to large datasets.

When $Y$ is discrete, recent works (Lipton et al., 2018; Azizzadenesheli et al., 2019) proposed to estimate $\beta = [\beta(y = 1), \ldots, \beta(y = K)]^T$ by using the confusion matrix of a classifier $f$:

$$\hat{\beta} = \tilde{C}\tilde{\beta}, \hspace{1cm}  (5)$$

where $\tilde{C}$ is the confusion matrix with each element $C_{ij} = \frac{1}{n_s} \sum_{k=1}^{n_s} \mathbb{1}\{f(x_k^s) = i, y_k^t = j\}$ and $\hat{\beta} = \frac{1}{n_t} \sum_{j=1}^{n_t} \mathbb{1}\{f(x_j^t) = i\}$. It can be seen that (5) corresponds to a specific form of (3) in which the feature mapping $\psi$ is set to $\psi(x) = \text{one-hot}(f(x))$, where $\text{one-hot}$ is a function mapping $y = f(x)$ to its corresponding one-hot vector. Because the dimensionality of $\psi(x)$ is simply the number of classes $K$, which is usually much smaller than the sample size, $\beta$ can be obtained efficiently. However, this type of methods only work for discrete labels.

### 3.2. Our Framework

Instead of estimating the density ratio $\beta(y)$, our framework estimates the target-domain marginal distribution using a constructed distribution $P^R_Y$ defined as follows:

$$P^R_Y = \int_{y^t} P_{X|Y}(x|y^t)P^R_Y(y^t)dy^t$$

$$= \int_{y^t} P_{X|Y}(x|y^t) \int_{y^s} P_{Y|R|Y^S}(y^r|y^s)P^S_Y(y^s)dy^sdy^r, \hspace{1cm}  (6)$$

where $P^R_Y$ is a form of $P^T_Y$ by transforming the training label distribution $P^S_Y$ using the transition model $\int_{y^r} P_{Y|R|Y^S}(y^r|y^s)P^S_Y(y^s)dy^s$. Because $Y$ is not observed in the test domain, we need to estimate the label transition model by comparing $P^R_Y$ and $P^T_X$. In the following sections, we will show how the transformation between $P^S_Y$ and $P^T_Y$ can be estimated from the labeled training set and unlabeled test set.

Figure 1 displays the flowchart of our framework. First, we transform the samples drawn from $P^S_Y$ using the Label Transformation network $LT$ which maps $P^S_Y$ to a distribution $P^R_Y$. Because there are only unlabeled data in the target domain, we cannot directly match $P^T_Y$ with the target-domain label distribution $P^R_Y$. Therefore, we then pass the transformed labels into the Label Influence Recovery network $G$, which models the conditional distribution $P_X|Y$ implicitly, to generate samples with distribution $P^R_X$. Finally, we match the generated distribution $P^R_X$ with the target domain $P^T_X$ to estimate the parameters in the label transformation network, such that $P^R_Y$ can approximate the target-domain label distribution $P^T_Y$. After estimating $P^R_Y$, we can train an unbiased classifier for prediction in the target domain. In the following, we present the details of each component in our framework.

#### 3.2.1. Label Transformation Network

Here we use a neural network $LT$ to transform the training label distribution $P^S_Y$ to a new label distribution $P^R_Y$, such that we can directly generate the corresponding sample distribution $P^R_X$ together with one generator $G$ that models $P_X|Y$. Specifically, we use the following functional model:

$$Y^R = LT(Y^S, Z), \hspace{1cm}  (7)$$

where $LT$ is modeled by a neural net and $Z$ is a random variable with distribution $P_Z$. (7) models the conditional distribution $P_X|Y^S$ implicitly. Because $P^R_Y = \int_{y^r} P_{Y|R|Y^S}(y^r|y^s)P^S_Y(y^s)dy^s$, we can sample $y^r_i \sim P^R_Y$ by first sampling $y^s_i$ from the source-domain labels, and then generate the corresponding $y^r_i = LT(y^s_i, z_i)$, where $z_i \sim P_Z$. Note that in some situations, such as discrete $Y$, it might be more convenient to directly use the parametric form of $P_{Y|R|Y^S}$.
If labeled data were available in the target domain, we can then simply match the empirical \( P_{X|Y} \) and \( P_Y \) to learn \( LT \). Unfortunately, target-domain labels are not available in unsupervised domain adaptation, but still, in the target domain we have unlabeled data \( \{x_t\}_{t=1}^{n_t} \), which can be used to estimate \( LT \). To this end, we need to transform \( P_Y \) to a distribution \( P_{Y|Y} \) in the \( X \) space. Because \( P_{X|Y} \) captures the influence of \( P_Y \), we can possibly estimate \( P_{X|Y} \) (or \( LT \)) by matching \( P_{X|Y} \) and \( P_X \), from which we can sample data points to estimate and minimize their distance.

### 3.2.2. LABEL INFLUENCE NETWORK

In order to transform \( P_{Y|Y} \) to \( P_{X,Y} \), we make use of the following model:

\[
X^R = G(Y^R, E),
\]

where \( G \) is a neural generator, and \( E \) is a random variable with distribution \( P_E \), which is set to normal distribution. We can use (8) to implicitly model \( P_{X|Y} \). Due to \( P_{X|Y} = \int_{y'} P_{X|Y|Y}(x_t|y') P_Y(y') \, dy' \), we can sample \( x_t^* \sim P_{X|Y} \) by first sampling \( y_t^* \) using (7), and then generate the corresponding \( x_t^* = G(y_t^*, e_t) \), where \( e_t \sim P_E \).

Since \( G \) corresponds to the generator in a conditional generative adversarial network (Mirza & Osindero, 2014; Miyato & Koyama, 2018; Gong et al., 2019), we can learn it from the source domain data \( D_s = \{x_t^s, y_t^s\}_{t=1}^{n_s} \) by adversarial training. Let \( Q_{X|Y} \) denote the conditional distribution specified by \( G \). If the input of \( G \) is drawn from \( P_{X|Y} \), the joint distribution of the generated data will be \( Q_{X,Y} = Q_{X|Y} P_{Y|Y} \). We can estimate \( G \) by minimizing the Jensen-Shannon Divergence (JSD) between \( Q_{X,Y} \) and \( P_{X|Y} \) (Mirza & Osindero, 2014):

\[
\begin{align*}
\min_{G} \max_{D_G} & \mathbb{E}_{(X,Y)\sim Q_{X,Y}} \left[ \log(D_G(X,Y)) \right] \\
& + \mathbb{E}_{E\sim P_E, Y\sim P_Y} \left[ \log(1 - D_G(G(Y,E), Y)) \right],
\end{align*}
\]

where \( D_G \) is an introduced discriminator (Goodfellow et al., 2014) to play the mini-max game together with \( G \). (9) is the negative cross entropy loss, and in some real experiments, we need to replace (9) by negative hinge-loss because it is more stable in image generation, as demonstrated in (Miyato et al., 2018; Brock et al., 2018).

### 3.2.3. DISTRIBUTION MATCHING

As described above, we can then construct a new data distribution \( P_{X} \), with the label transformation network \( LT \) and the label influence network \( G \). To estimate \( LT \), we fix \( G \) and minimize the JSD between \( P_{X} \) and \( P_{X} \) w.r.t. \( LT \) by the following objective:

\[
\min_{LT} \max_{D_{LT}} \mathbb{E}_{D_{LT}} \left[ \log(D_{LT}(X)) \right] + \mathbb{E}_{Z\sim P_Z, E\sim P_E, Y\sim P_{Y|Y}} \left[ \log(1 - D_{LT}(G(LT(Y^S, Z), E))) \right].
\]

where \( D_{LT} \) is an introduced discriminator (Goodfellow et al., 2014) to perform adversarial training with \( LT \). In detail, when the label is continuous, the whole network is totally differentiable, so we can simply estimate \( LT \) by using backpropagation. However, in the case of discrete labels, we cannot backpropagate through the label \( y_t^* \). Fortunately, we can assume a parametric form of \( P_{X} \), i.e., the categorical distribution, in case of discrete \( Y \). Thus, we can make use of the Gumbel-softmax trick (??) or the REINFORCE trick (Williams, 1992) to backpropagate through the discrete labels \( y_t^* \). The two tricks have been successfully employed in various problems such as text generation (Yu et al., 2017a) and neural architecture search (??).

**Gumbel-softmax Trick** Let \( Y^{Ro} \) and \( Y^{So} \) denote the one-hot representations of \( Y^R \) and \( Y^S \), respectively. We can use a special \( LT \) function to sample from \( P_{Y|Y|Y} \):

\[
\hat{Y}_k^{Ro} = \frac{\exp((\log M_k Y^{So} + Z_k)/\tau)}{\sum_{i=1}^{K} \exp((\log M_i Y^{So} + Z_i)/\tau)},
\]

where \( \hat{Y}_k^{Ro} \) is the \( k \)th element of \( \hat{Y}^{Ro} \), \( Z_k \sim Gumbel(0, 1) \), \( \tau \) is the temperature, and \( M_k \) is the \( k \)th row of the transition matrix \( M \), whose \( i \)th element is \( P(Y^R = i|Y^S = j) \). As \( \tau \to 0 \), \( \hat{Y}^{Ro} \) provides a good approximation of the one-hot \( Y^{Ro} \). Since the softmax function is differentiable, it
enables end-to-end learning of $LT$, which only contains $M$ as parameters.

**REINFORCE Trick** Because $P_{Y^R | Y^S}$ involves learnable parameters $M$, we rewrite it as $P_M(Y^R | Y^S)$ and reformulate (10) as

$$\min_M \max_T \mathbb{E}_{X \sim P_X^Y} \left[ \log(D_{LT}(h(X))) \right] + \mathbb{E}_{E \sim P_E, Y^R \sim P_{M(Y^R | Y^S)}, Y^S \sim P_Y^S} \left[ \log(1 - D_{LT}(G(Y^R, E))) \right].$$

The gradient w.r.t. $LT$ can be written as:

$$\nabla_M \log P_M(Y^R | Y^S).$$

**Feature Matching** Generally speaking, we estimate the prior distribution in the target domain $P_Y^T$ by comparing the marginal distributions of $X$ in the target domain and the transformed source domain. However, for some high dimensional data such as images, $X$ might contain many redundant features $X_R$ that are unrelated to $Y$, causing unnecessary estimation errors of $P_Y^T$. Intuitively, this is because the conditional distributions of these redundant features $X_R$ satisfy $P_{X_R | Y} = P_{X_R}$, which are not helpful in identification of $P_Y^T$ but will cause additional estimation error. To improve the estimation accuracy, we propose to estimate $LT$ by matching $P_{h(X)}^R$ and $P_{h(X)}^T$, instead, where $h$ is a pre-trained network that extracts compact representations from raw $X$ data. Therefore, we replace (10) by

$$\min_T \max_D \mathbb{E}_{X \sim P_X^Y} \left[ \log(D_{LT}(h(X))) \right] + \mathbb{E}_{Z \sim P_Z, E \sim P_E, Y^S \sim P_Y^S} \left[ \log \left(1 - D_{LT}(G(LT(Y^S, Z), E)))\right) \right].$$

Ideally, we aim to find $h(X)$ such that $Y \perp \perp X|h(X)$ by using the source-domain labeled data. This conditional independence property implies that $h(X)$ contains all information in $X$ that is relevant to $Y$. Learning conditional invariant representation has been shown to be effective in correcting covariate shift (Stojanov et al., 2019). However, since Stojanov et al. (2019) set $h$ as a linear transformation and measure conditional dependency using kernel measures (Fukumizu et al., 2004), the method cannot learn compact representations for images and is computationally expensive. Here we use a convolutional network as $h$ to extract feature representations and measure the dependency by assuming a (generalized) linear model for $P_{Y|h(X)}$. This is sensible because the features extracted by nonlinear neural networks are usually linearly separable. Proposition 1 shows how $h$ can be learned to satisfy the conditional independence property. (The proof can be found at the supplementary material A.1)

**Proposition 1** Assuming $P_{Y|h(X)}$ can be modeled by a (generalized) linear model, i.e., linear regression model for continuous $Y$ and multinomial logistic regression model for discrete $Y$. Let sample size $n \to \infty$, $h$ learned by minimizing the mean squared error (for continuous $Y$) or the cross-entropy loss (for discrete $Y$) satisfies $Y \perp \perp X|h(X)$.

### 3.2.4. Shift Correction

After quantifying the target label distribution $P_Y^T$, the model with target shift problems should be corrected and adapted to the target domain. The previous work choose to re-train the model under the importance-weighted ERM framework (Gretton et al., 2009; Shimodaira, 2000; Sugiyama et al., 2008). In our framework, we can retrain the source-domain model with new data drawn from our model. As it is time-consuming to retrain a new model, a quick adaptation method is also provided in our framework. As described by Proposition 1, if $h$ learned at the uniform Training set satisfies the conditional independence property with $Y$, the output layer of a neural network is the only module needed to be adapted to the new label distribution $P_Y^T$ given the feature extractor $h$. In our framework, we fine-tune the output layer several epochs using the samples generated by our Label Influence Recovery network $G$ with the label distribution $P_Y^T$ learned by Label Transformation Network $LT$. As such, the output layer will be quickly adapted to the target domain.

### 4. Experiments

To verify the effectiveness and universality of the proposed framework, we design experiments for three target shift scenarios, i.e., the discrete, continuous, and multi-dimensional target shift, on various datasets.

#### 4.1. Discrete Target Shift Experiments

We compare our method with the competitors on three datasets, e.g., MNIST, FASHION-MNIST, and CIFAR10 (Krizhevsky & Hinton, 2009). We follow the same setting of BBSE (Lipton et al., 2018) and RLLS (Azizzadenesheli et al., 2019). Specifically, for MNIST, we use a simple two-layer neural network; the Resnet-18 (He et al., 2016) and CNN in DCGAN (Radford et al., 2015) are chosen for CIFAR 10 and FASHION-MNIST, respectively. The learning rate is set to 0.01. Moreover, we use the network architecture of BigGAN (Brock et al., 2018) and the loss of TAC-GAN (Gong et al., 2019) to model the invariant conditional distribution $P_{X|Y}$. The original training sets given in the datasets are used as the training set for the proposed method and the baselines. The test set is sampled to have a specific label distribution $P_Y^T$ and is of size 10,000. For the quantification of $P_Y^T$, we use the REINFORCE trick instead of Gumbel-softmax trick, as the temperature $\tau$ in
the Gumbel-softmax trick is hard to choose.

4.1.1. Shift Settings

In our paper, the label distribution $P_{Y}^{S}$ in the training set is a uniform distribution over all classes. For the test set, we consider three types of shifts: Tweak-One shift, Minority-Class shift, and Random Dirichlet shift. These settings are designed to capture diverse label probability changes, i.e., large label probability change, small label probability change, and random label distribution change. We repeat the experiments 10 times to verify the effectiveness and robustness of the proposed method.

**Tweak-One Shift** To evaluate the performance on the large label probability quantification. In our experiments, the ratio of one class is set to $[0.5, 0.6, 0.7, 0.8, 0.9]$, respectively, while ratios of other classes are uniform.

**Minority-Class Shift** To evaluate the performance on the small label probability quantification. In our experiments, $[20\%, 30\%, 40\%, 50\%]$ classes are set to 0.001, respectively, while ratios of other classes are uniform.

**Random Dirichlet Shift** In this shift, we randomly generate a label distribution $P_{Y}^{T}$ by employing the Dirichlet distribution with different values of the concentration parameter $\alpha$. Then, we re-sample the test set according to the generated distribution $P_{Y}^{T}$. In our experiments, $\alpha$ are set to 10, 1, 0.1, 0.01. Note that the generated label distribution $P_{Y}^{T}$ tends to be smoother for bigger $\alpha$.

4.1.2. Evaluation Metrics and Results

As done in BBSE (Lipton et al., 2018) and RLLS (Azizzadenesheli et al., 2019), the accuracy and F1 score (Goulette & Gaussier, 2005) are used as evaluation metrics, allowing us to compare the performance of different methods more comprehensively (Azizzadenesheli et al., 2019). We also evaluate the estimation error of the estimated label weights ($P_{Y}^{T} / P_{Y}^{S}$) by using mean square error (MSE).

We compare our method with the two recent methods: BBSE (Lipton et al., 2018) and RLLS (Azizzadenesheli et al., 2019), which estimate label weights $\beta$ using the confusion matrix of a classifier $f$ trained on the training set. To verify Proposition 1, we consider a variant of RLLS called RLLS(feature), which matches distributions on the feature space $h(X)$. RLLS(feature) can also be considered as setting $\psi(X)$ to $h(X)$ in (3). For the evaluation of the shift correction, we evaluate the performance of classifiers trained on the training set without adaptation (denoted as Baseline) and the classifiers trained on weighted training sets, where the weights are estimated by using target domain labels (denoted as BEST(ERM)). Similarly, we also test the classifiers trained on weighted training data, where the weights are obtained by RLLS, BBSE and RLLS(feature). For our method, we have two ways to utilize the label distributions estimated by our framework. The first one is to re-train a new classifier using the weighted training set (Ours(ERM)). The second one is the fine-tuning method described in 3.2.4, which is denoted as Ours(Fine-tune). Specifically, we fine-tune the output layer of the pretrained classifier on the source domain by 10 epochs, using the data generated from our model.

Due to the page limit, we only show the results of CIFAR10 dataset in the paper and the results of MNIST and FASHION-MNIST can be found in the supplemental materials A.2. In terms of the estimation error of the target label distribution $P_{Y}^{T}$, the subfigure (a) of Figure 2, 3, 4 demonstrate that the label weights estimated by our framework are more accurate and stable than previous methods. In addition, the RLLS (feature) algorithm that matches label distribution on feature space of classifier trained on the training set also achieves better performance than BBSE and RLLS in most settings. For the accuracy and F1 score of the corrected classifiers, subfigures (b) and (c) of Figure 2, 3, 4 show that the classifier corrected by our framework can achieve better performance in both two evaluation metrics in most settings. Also, our fast fine-tune method achieves comparable performance with re-weighting methods.
4.2. Continuous Target Shift Experiments

In this section, we design two experiments to verify the effectiveness of our model on continuous target shift problems. Firstly, we conduct experiments on a synthetic data that evaluates the performance of our framework on simple continuous target shift problems. Then we apply our model on a real data application: Object 1D position prediction (Matthey et al., 2017), which evaluates the performance of our framework on simple continuous target shift problem in the high-dimensional X situation.

4.2.1. SYNTHETIC DATA EXPERIMENT

In this experiment, we design a toy dataset by modifying a classic and popular synthetic data experiment (MOON dataset (Ganin et al., 2017)) in covariate shift. We first generate two quarter circles with radius $R$ 10 and sample size 1000 as the training set, which is shown in Figure 5(a). The range of a single continuous label is from -10 to 10 and the values are uniformly distributed. Then we generate the test set with 500 samples in the same way according to several target label distributions. Here we consider 4 types of target shift to evaluate model performance and robustness.

**Experimental Setting** In this experiment, the architectures of all modules in our framework are three-hidden layers neural networks with 10 hidden neurons. In the distribution matching module, the baseline KMM (Zhang et al., 2013) uses MMD with the median kernel width to match the built data distribution $P^Y_{X_{new}}$ and target data distribution $P^Y_{X_{new}}$. To fairly compare the methods, we also use MMD to do distribution matching.

**Shift Settings** To evaluate the model’s label quantification performance, we set 4 target shift situations. **Shift A:** Set the target label distribution $P^Y_{X_{new}}$ as a Gaussian distribution with the mean of $\frac{\sqrt{2}}{2} R$ and variance of 1. **Shift B:** Set the target label distribution $P^Y_{X_{new}}$ as a Gaussian distribution with the mean of $-\frac{\sqrt{2}}{2} R$ and variance of 1.

**Shift C:** The target label distribution is a mixture Gaussian distribution with Shift A and Shift B, with a mixture proportion 0.5. **Shift D:** The target label distribution is a random label distribution generated by a randomly parameterized neural network.

**Baselines** The classic KMM methods (Zhang et al., 2013) are chosen as our baselines. We consider two variants: KMM that matches the distributions in the raw input space and KMM(feature) that matches the distributions in feature space of the regressor.

![Figure 4](image-url) (a) Mean squared errors of estimated label weights (lower is better), (b) accuracy, and (c) F-1 score (higher is better) on CIFAR10 for uniform training set and minority-class shifted test set, where alpha is the ratio of minority classes.

![Figure 5](image-url) (a) The illustration of Moon Synthetic Data (Shift C), where the generated two quarter circles training set as blue symbols show. (b) The visualization of label weight $P^Y_{X_{new}}$ of KMM, KMM(feature), our framework and the Ground Truth.

<table>
<thead>
<tr>
<th></th>
<th>SHIFT A</th>
<th>SHIFT B</th>
<th>SHIFT C</th>
<th>SHIFT D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.0061</td>
<td>0.0059</td>
<td>0.0055</td>
<td>0.034</td>
</tr>
<tr>
<td></td>
<td>±0.0012</td>
<td>±0.0008</td>
<td>±0.0009</td>
<td>±0.0114</td>
</tr>
<tr>
<td>KMM</td>
<td>0.0048</td>
<td>0.0044</td>
<td>0.0044</td>
<td>0.0275</td>
</tr>
<tr>
<td></td>
<td>±0.0011</td>
<td>±0.0005</td>
<td>±0.0004</td>
<td>±0.0096</td>
</tr>
<tr>
<td>KMM (feature)</td>
<td>0.0045</td>
<td>0.0039</td>
<td>0.0043</td>
<td>0.0276</td>
</tr>
<tr>
<td></td>
<td>±0.0007</td>
<td>±0.0006</td>
<td>±0.0004</td>
<td>±0.0097</td>
</tr>
<tr>
<td>Ours</td>
<td>0.0036</td>
<td>0.0024</td>
<td>0.0036</td>
<td>0.0251</td>
</tr>
<tr>
<td></td>
<td>±0.0002</td>
<td>±9e-5</td>
<td>±0.0004</td>
<td>±0.0121</td>
</tr>
</tbody>
</table>

Table 1. The results of Continuous target shift Synthetic Data Experiments. The value is the mean square error of prediction value and groun truth. The baseline is the original regressor trained on the standard training set, and the KMM is (Zhang et al., 2013).

**Results** In this section, to compare the performance of estimated target label distribution qualitatively, we visualize the estimated density ratio $(P^Y_{X_{new}} / P^Y_{X_{new}})$ of Shift C in Figure 5. More figures about other shift settings can be found in supplementary materials A.3 Visually, our model has better label distribution estimation performance compared with other methods.

Then we evaluate the mean square error of the baseline regressor without adaptation and three others corrected.
by KMM, KMM(feature), and Ours(Adv) respectively. The results are shown in Table 1. It can be seen that the MSE errors of our framework are significantly lower than those of the other methods in all shift settings, and KMM (feature) achieves slightly better performance than the original KMM method in some settings, which verifies the correctness of Proposition 1.

4.2.2. OBJECT 1D LOCATION PREDICTION EXPERIMENT

In this experiment, we use a popular disentanglement dataset called Sprites [1] (Matthey et al., 2017). This dataset consists of 737,280 2D shapes images, which are generated from 6 ground-truth independent latent factors. Some example images are shown in Figure 6. The factors include color, shape, scale, rotation, x, and y positions of a sprite. We choose the x or y position of sprites as the target variable and consider it as a regression problem.

Figure 6. Illustration of the sprites dataset. This sprites in this dataset have 3 shapes(square, ellipse, heart), 6 scales values linearly spaced in [0.5, 1], 40 orientation values in [0, 2 pi], 32 X position values in [0, 10], 32 Y position values in [0, 10]

Figure 7. The prediction mean square error of 1D sprite position prediction (lower is better). (a) Random Dirichlet shift, where the smaller alpha corresponding to the bigger shift. (b) Large target shift, where alpha is the possibility of shifted label. (c) Minority shift, where alpha is the ratio of minority classes.

Experimental Setting We use the network architecture in DCGAN (Radford et al., 2015) as feature extractor for the regressor, the learning rate for the regressor is set to 1e-4, which is the best learning rate according to our experiments. The DCGAN (Radford et al., 2015) is used to model the invariant distribution $P_{X|Y}$ and the Transformation Model $LT$ is a simple 3-layer neural network.

For training set, we randomly sample 40000 images with uniform x value distribution from overall dataset and the test set consists of 40000 samples (sampled from specified distribution for target shift).

4.3. Multi-Dimensional Target Shift Experiments

In this experiment, we design a simple multi-dimensional target shift experiment, which is object 2D location prediction. We use the same dataset with the object 1D location prediction experiment, but we predict both x and y position values of a sprite. As such, the label $Y$ in this experiment is 2-dimension, increasing the difficulty of detecting and correcting the target shift.

Experimental Setting We use the same network architecture and train/test split as described in 4.2.2. For Transformation Model $LT$, two networks are used to model the x and y position target shift respectively as the x and y position value in this dataset are independent. As such, using two networks will reduce the difficulty of quantifying target label distribution $P_{Y|T}$.

Shift Settings The settings are also same with 4.2.2 described, but we shift the x and y position value respectively.

Baselines and Results Similar to the 1D position prediction, the baseline is the baseline regressor without adaptation as our methods is the first method which is compatible with large-scale multi-dimensional target shift problems. The results are shown in Figure 8. It can be seen that the regressor corrected by our framework can
achieve lower MSE error than the baseline method in most settings.

5. Conclusion

In this paper, we propose an end-to-end target shift quantification and correction framework called Label Transformation Framework which can deal with discrete, continuous and multi-dimensional target shift problems. Based on this framework, we further find that matching the distributions of a feature representation of $X$ that discards the information irrelevant to $Y$ can have better performance over other methods which quantify the label distribution $P^T_Y$ based on scratch data or biased output. In the experiments, we apply our framework to several classification and regression tasks under various target shift settings. The results show that our framework has better performance and universality than previous methods. Future work will be extending our framework to address conditional shift, where $P_{X|Y}$ also changes across domains.

6. Acknowledgements

This work was supported by Australian Research Council Projects FL-170100117, DP-180103424, IH-180100002, IC-190100031, DE-190101473, LE-200100049 and the United States Air Force under Contract No. FA8650-17-C-7715.

References


LTF: A Label Transformation Framework for Correcting Target Shift


Zachary C. Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors, 2018.


Petar Stojanov, Mingming Gong, Jaime Carbonell, and Kun Zhang. Low-dimensional density ratio estimation for


