5-2022

Robust and Fair Machine Learning under Distribution Shift

Wei Du
University of Arkansas, Fayetteville

Robust and Fair Machine Learning under Distribution Shift

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering with a concentration in Computer Science

by

Wei Du
Northeastern University
Bachelor of Engineering in Electronics and Information Engineering , 2012
Zhejiang University
Master of Science in Electronic Science and Technology, 2015

May 2022
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Xintao Wu, Ph.D.
Dissertation Director

Brajendra Panda, Ph.D.
Committee member

Qinghua Li, Ph.D.
Committee member

Zhenghui Sha, Ph.D.
Committee member

ABSTRACT

Machine learning algorithms have been widely used in real world applications. The development of these techniques has brought huge benefits for many AI-related tasks, such as natural language processing, image classification, video analysis, and so forth. In traditional machine learning algorithms, we usually assume that the training data and test data are independently and identically distributed (iid), indicating that the model learned from the training data can be well applied to the test data with good prediction performance. However, this assumption is quite restrictive because the distribution shift can exist from the training data to the test data in many scenarios. In addition, the goal of traditional machine learning model is to maximize the prediction performance, e.g., accuracy, based on the historical training data, which may tend to make unfair predictions for some particular individual or groups. In the literature, researchers either focus on building robust machine learning models under data distribution shift or achieving fairness separately, without considering to solve them simultaneously.

The goal of this dissertation is to solve the above challenging issues in fair machine learning under distribution shift. We start from building an agnostic fair framework in federated learning as the data distribution is more diversified and distribution shift exists from the training data to the test data. Then we build a robust framework to address the sample selection bias for fair classification. Next we solve the sample selection bias issue for fair regression. Finally, we propose an adversarial framework to build a personalized model in the distributed setting where the distribution shift exists between different users.

In this dissertation, we conduct the following research for fair machine learning under distribution shift.

- We develop a fairness-aware agnostic federated learning framework (AgnosticFair) to deal with the challenge of unknown testing distribution;
- We propose a framework for robust and fair learning under sample selection bias;
- We develop a framework for fair regression under sample selection bias when dependent variable values of a set of samples from the training data are missing as a result of another hidden process;
- We propose a learning framework that allows an individual user to build a personalized model in a distributed setting, where the distribution shift exists among different users.

ACKNOWLEDGEMENTS

The Ph.D. study in past few years is an unforgettable journey in my life, from which I have gained plenty of growth and achievements. I am so fortunate to receive many help and support from many ways and express my deep appreciation for all of them.

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Xintao Wu, for his time, guidance, support, and patience throughout the period of my Ph.D. study. Dr. Wu has provided not only invaluable suggestions for my research, but also shared the wisdom of life which helped me to go through the toughest times. His excellency and enthusiasm as a researcher will continue motivating me in my future career.

I would like to thank my dissertation committee members, Dr. Qinghua Li, Dr. Brajendra Panda, and Dr. Zhenghui Sha. Their constructive comments significantly improved the quality of this dissertation.

I would also like to thank Dr. Hanghang Tong from UIUC and Dr. Aidong Lu from UNC Charlotte, with whom I have been truly fortunate to collaborate throughout my Ph.D. journey. Their advice, experience, and ideas are vital in my research projects.

I would also like to thank my friends and colleagues in the Social Awareness and Intelligent Learning Lab at the University of Arkansas. I appreciate the collaboration with Dr. Depeng Xu, and Minh-Hao Van. Thanks also go to the other friends and colleagues who gave me consistent encouragement: Dr. Lu Zhang, Dr. Shuhan Yuan, Dr. Panpan Zheng, Dr. Yongkai Wu, Wen Huang, Dr. Kevin Labille, Vinay MS, Alycia Carey, Aneesh Komanduri, Karuna Bhaila, and Huy Mai. I have been truly honored to work with these excellent researchers.

Finally, I would express my deepest gratitude to my family for their unconditional

love and support. This dissertation is dedicated to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

In this chapter, we introduce the motivation and provide an overview of this dissertation, and then summarize the contributions of this research.

1.1 Motivation

Machine learning algorithms have been widely used in real world applications. The development of these techniques has brought huge benefits for many AI-related tasks, such as natural language processing [1, 2, 3], image classification [4, 5], video analysis [6], and so forth. In traditional machine learning algorithms, we usually assume that the training data and test data are independently and identically distributed (iid), indicating that the model learned from the training data can be well applied to the test data with good prediction performance. However, this assumption is quite restrictive because the distribution shift can exist between the training data and test data in many scenarios. For instance, the collection of medical results may be only available from some certain groups as the data from other groups are very difficult and costly to obtain. Sensor data collection can be another example. It is natural that different sensors may malfunction at different rates or we collect data with different rates, indicating that some portions of the observed data are either under-represented or over-represented. These scenarios make the iid assumption inappropriate and incorrect, which can impose huge challenges to construct useful and reliable machine learning models that could work in practice.

Another issue in traditional machine learning is the potential discrimination in the model's decision output. The goal of traditional machine learning model is to maximize the prediction performance, e.g., accuracy, based on the historical training data. As a conse-

quence, such models may tend to make unfair and discriminative predictions for some particular individuals or groups. One particular example is Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) [7], an machine learning algorithm applied in multiple USA states to determine how likely a defender will be re-offended in the future. A detailed analysis from ProPublica shows that the risk score computed by the algorithm is more biased to the black defender, e.g., COMPAS is almost twice as likely to incorrectly determine black defendants as high risk than white defendants. Therefore, unfairness and bias caused by the machine learning models can cause serious social concerns and damage to some unfavorable groups. It is imperative to develop fairness-aware machine learning algorithms so that the decision can be made fair and reliable to all groups of people.

The issue of distribution shift in machine learning has drawn increasing attentions in the past few years. When facing the distribution shift between the training and test data, we first need to characterize the types of changes that occur from the training data to the test data, which can help in determining the appropriate techniques to solve this issue. Researchers have divided the distribution shift into several cases [8], such as covariate shift, prior probability shift, domain shift, and so forth. Various techniques have been developed to solve the distribution shift based on different scenarios. Taking the covariate shift as an example, weighted sample [9] is proposed to correct the distribution shift, where we assign different weights for the training data to resemble the test data so that the learned model can be applied to the test data with good predictive performance. In addition, how to detect and calibrate prior probability shift can be found in [10], and adversarial learning approaches to solve domain shift are also reported [11, 12].

Fairness-aware machine learning is also an active area and researchers have developed approaches to unveil the discrimination by either investigating historical data or prediction

results, and then achieve fairness by modifying the biased data, or enforcing constraints for the models, or processing unfair prediction results. Different fairness notions [13] are also proposed to quantify the strength of unfairness, such as demographic parity, equality of odds and equality of opportunity. Based on these notions, various frameworks are designed to achieve the fairness. For example, Hardt et al. [14] adjust a trained unconstrained model to remove discrimination based on equalized odds. After adjustment, the unconstrained model behaves like a randomized classifier that assigns each data point a probability conditional on its protected attribute and predicted label. These probabilities are calculated by a linear program to minimize the expected loss. Moreover, different approaches to ensure different fairness notions are also reported [15, 16, 17].

However, most of the studies only focus on correcting distribution shift between the training and test data or ensuring fairness of the machine learning models separately, without considering the connection between them. However, there are several challenges in achieving fair machine learning under distribution shift. First, how to correct the distribution shift and achieve fairness simultaneously is under explored. Second, directly applying the approach to solve distribution shift may cause more unfairness in machine learning models. Third, there exist many different fairness notions and we may need to achieve several fairness notions simultaneously, therefore it is necessary to design a unified framework which is able to incorporate different fairness notions under distribution shift.

1.2 Overview

The goal of this dissertation is to address challenging issues in fairness-aware machine learning under distribution shift.

First, we focus on how to achieve fairness under distribution shift in the federated

learning setting. Federated learning [18] is an attractive framework to handle complex data [18, 19, 20] as it enables a new implementation of distributed deep learning over a large number of clients. Although tremendous research has been done on the federated learning, how to achieve fairness in federated learning is under-explored. One big challenge in federated learning is that the training data is geo-distributed and the distribution of different clients is also different. In addition, the test data distribution may also be unknown which triggers another challenge of developing fair federated learning algorithms. Directly applying fairness constraint on the training data in federated learning cannot guarantee fairness on the test data. To address the above challenges, we propose fairness-aware agnostic federated learning to achieve both good prediction performance and fairness for unknown test data (Chapter 4).

Second, we focus on how to achieve fairness in machine learning under sample selection bias. In traditional supervised learning, the underlying assumption is that the training data and test data are drawn from the same distributions. However, when the distributions on training and test data sets do not match, we are facing sample selection bias. Consequently, the learned model on the biased data is more vulnerable and will incur more accuracy loss and unfairness when it comes to the test data. We investigate this problem by adopting reweighing estimation approach for bias correction and minimax robust estimation to build a robust framework so that the learned machine learning models are both fair and robust for the test data (Chapter 5).

Third, we explore on fairness-aware regression when it comes to the sample selection bias. Much of existing work has focused on the fair classification, while fair regression has received less attentions. The prediction outcome of regression is continuous which is quite different from classification, and new fairness notions are needed to quantify the unfairness in

regression so that previous frameworks on fair classification may not be applied. In addition, the possible sample selection bias poses another challenge to build fairness-aware regression models. We propose a general framework for fair regression under sample selection bias, where different fairness notions can be incorporated into this framework (Chapter 6).

Fourth, we study how to select similar data for an individual user and build a personalized model in a distributed setting. The traditional approach to build a machine learning model is to first collect data from different sources and then build a global model on these data. However, the global model may not be optimal for individual users as the data distribution from different users may vary. The issue is that an individual user usually can collect limited data and needs to retrieve similar data from other users, which brings the challenge that how to efficiently select similar data. In addition, with the retrieved similar data, another challenge is how to better improve the performance of the personalized model for individual users. We design an adversarial framework that allows an individual user to efficiently select similar data and build a personalized model (Chapter 7).

The remainder of this dissertation is organized as follows. In Chapter 2, we discuss related work on how to correct distribution shift in machine learning and develop different frameworks to ensure fairness in machine learning. Then in Chapter 3, we present some preliminary background for distribution shift and fairness-aware machine learning. The main body of this dissertation is in Chapters 4 - 7. Finally, we conclude this dissertation and discuss future work in Chapter 8.

1.3 Summary of Contributions

In Chapter 4, we propose fairness-aware agnostic federated learning (AgnosticFair) to deal with the challenge when the testing data distribution is unknown. We formulate

AgnosticFair as a two-player adversarial minimax game between the learner and the adversary. The adversary aims to generate any possible unknown testing data distribution to maximize the classifier loss. We assign an individual reweighing value on each training sample and incorporate reweighing value in both agnostic loss function and agnostic fairness constraint. As a result, the global model learned in the minimax game achieves both high accuracy and fairness guarantee on unknown testing data. Moreover, each client can simply deploy the global model on its local site as the learned global model guarantees fairness on any local data. We conduct extensive experiments on two public datasets and compare our approach with several baselines. Evaluation results demonstrate the effectiveness of our proposed AgnosticFair in terms of accuracy and fairness under data shift scenarios. To our best knowledge, we are the first one that formulates the problem of fairness-aware federated learning under the data distribution shift among the clients.

In Chapter 5, we develop a framework for robust and fair learning under sample selection bias. We embrace the uncertainty incurred by sample selection bias by producing predictions that are both fair and robust in test data. Our framework adopts the reweighing estimation approach for bias correction and the minimax robust estimation approach for achieving robustness on prediction accuracy. Moreover, during the minimax optimization, the fairness is achieved under the worst case, which guarantees the model's fairness on test data. To address the intractable issue, we approximate the fairness constraint using the boundary fairness and combine into the classifier's loss function as a penalty. The modified loss function is minimized in view of the most adverse distribution within a Wasserterin ball centered at the empirical distribution of the training data. We present two algorithms, $RFLearn^1$ for the scenario where the unlabeled test dataset $\mathcal{D}$ is available, and $RFLearn^2$ for the scenario where $\mathcal{D}$ is unavailable. In $RFLearn^1$, we estimate the sample selection

probability via its density ratio of training data and test data and then correct the bias in loss function. In *RFLearn*^{2}, we introduce some natural assumptions, i.e., the samples in the same cluster have the same selection probability which is within a range from the uniform selection probability. The algorithm first clusters the training data and robustifies the sample selection probability estimation of each cluster within a Wasserterin ball. We test our algorithms on two real-world datasets and experimental results demonstrate that our algorithms can achieve both good performance on prediction and fairness.

In Chapter 6, we propose, *FairLR**, the fair regression framework under sample selection bias when dependent variable values of a set of samples from the training data are missing as a result of another hidden selection process. Our *FairLR** adopts the classic Heckman model [21] for bias correction and the Lagrange duality theory [22] to achieve regression fairness based on a variety of fairness notions. Our fair regression framework minimizes the loss function subject to fairness inequality and equality constraints. We apply the Lagrange duality theory to transform the primal problem into a dual convex optimization problem and analyze whether fairness metrics satisfy the Slater condition, thus achieving strong duality. For the two popular fairness notions, mean difference (MD) and mean squared error difference (MSED), we further derive two explicit formulas without optimizing iteratively. We conduct experiments on three real-world datasets and the experimental results demonstrate our approach's effectiveness in terms of both utility and fairness.

In Chapter 7, we develop a learning framework that enables an individual user to effectively collect data and build a robust model on the combination of the collected data and his own data. For data collection, we propose an approach of using an auto-encoder and a generative adversarial network (GAN) [23]. The auto-encoder is used to obtain data representation that is further used to train the GAN. The trained encoder and the discrim-

inator from GAN are sent to his neighbors. Each neighbor user uses the encoder to obtain the representation of his data and then uses the discriminator to calculate the probability score of his data. Data with high probability scores are combined with the user's original data to train the personalized model. The advantage of using GAN is that it can capture inherent properties of the underlying data without manually specifying features. With the requested data, we develop two approaches to improve the performance of the personalized model. The first approach is weighted learning by assigning a different weight to each record of the requested data. The data record with a high probability score computed by the discriminator is assigned with a high weight. The weighted learning is able to capture the importance of different data. The second approach is the adversarial learning that aims to minimize the distribution discrepancy between the requested data and user's own data. The core idea of the adversarial learning is to map both the requested data and user's own data into the same feature space where the distribution discrepancy is minimized. Our adversarial training is analogous to the discriminator of the GAN. The role of the discriminator is to predict whether the generated features are from user's own data or the requested data.

2 Related Work

2.1 Fair Machine Learning

Fairness-aware learning received lots of attentions in the past few years. The goal of fair machine learning is to remove discrimination and unfairness by either changing the biased data and/or the predicted models built on the biased data. There have been lots of works in past few years in achieving fairness in machine learning systems, which can be generally divided into three groups: pre-processing, in-processing and post-processing.

The mainstream of pre-processing approaches [24, 25, 26, 27] is to modify the training data to remove discriminatory patterns. The logic is that the machine learning model will make fair predictions if it is trained on discrimination-free data. Several approaches [26, 28] are proposed to modify the training data, including massaging, which modifies the labels of some individual data records, sampling, which changes the sample size of different groups, reweighting, which assigns different weights to individuals. In [27], the authors investigate on the removal of indirect discrimination from the original data. In particular, they modify the information of the non-sensitive attributes so that we cannot infer any information of the sensitive attributes. Calmon et al. [29] propose a probabilistic formulation of data pre-processing for reducing discrimination, where a convex optimization is designed to control discrimination while limit distortion in individual data samples. Zhang et al. [16] develop a causal graph based approach to remove the discrimination. The research in [30] applies generative adversarial neural networks to generate fair data from the original training data and uses the generated data to train the model. In addition, learning fair representations from the training data also belongs to the pre-processing approaches. For example, Zhao et

al. [31] investigate the relationship between fair representation and decision making with theoretical guarantee. They employ the adversarial technique to extract the fair representation. Song et al. [32] propose an information theoretically motivated objective to learn maximally expressive representations subject to fairness constraints.

The core idea of in-processing is to incorporate the fairness constraint into the classification model during the optimization process [15, 33, 34, 35, 36, 37, 38]. For instance, the work [37] adds a non-discrimination constraint on the training samples during the optimization of the classifier. The authors in [35] study the optimization of non-convex and non-differentiable constraints induced by the fairness. Zafar et al. [15] introduce a flexible mechanism to achieve fairness in machine learning. They leverage a novel intuitive measure of decision boundary fairness and transform it into a convex optimization, and the proposed approach can be applied into logistic regression, linear regression and SVM. Wu et al. [39] transform the fairness constraints to be convex and provide theoretical guarantees to achieve fairness for classifiers. Donini et al. [34] present an empirical risk minimization to incorporate fairness constraint into the learning problem. It aims to make the conditional risk of the classifier to be approximately constant subject to the sensitive variable. Both risk and fairness bounds are derived to support the statistical consistency.

Post-processing approaches modify the prediction label by the trained classifiers. In fact, some of the pre-processing methods can also be applicable into the post-processing, such as massaging, reweighting, and sampling. There are also lots of works developed specifically for post-processing. For example, Hardt et al. [14] adjust a trained unconstrained model to remove discrimination based on equalized odds. After adjustment, the unconstrained model behaves like a randomized classifier that assigns each data point a probability conditional on its protected attribute and predicted label. These probabilities are calculated by a linear

program to minimize the expected loss. Kim et al. [40] propose a rigorous framework of post-processing to achieve fairness across different subgroups. They assume black box access to the predictor and a relatively small set of labeled data for auditing and leverage the boosting mechanism to remove the bias.

2.2 Distribution Shift in Machine Learning

Distribution shift widely exists in machine learning as the data in which we use during the training will differ from what we face in the future test scenario. In many cases, researchers may ignore the differences by presuming either the training and test data match, or it makes no difference if they do not match. In fact, the mismatch can cause the performance degradation or incorrect predictions when deploying the trained models into practice. There have been lots of works in the past years to develop different frameworks to correct the distribution shift and make the machine learning models more robust.

The earlier works to correct distribution shift, especially covariate shift, focus on the density ratio estimation. For covariate shift, we usually assume that the feature distribution $P(X)$ changes over training and test data, while the conditional distribution $P(Y|X)$ from label to data remains unchanged, where X is the data feature and Y denotes the data label. One major direction is to estimate the weight between the training and test data. For example, Sugiyama et al. [41] estimate the input densities of the training and test data and then estimate the importance by the ratio of the density estimates. The authors prove that the reweighing of each instance is asymptotically optimal for log-likelihood estimation. In [9], Cortes et al. develop a cluster based technique to estimate the reweighing function from the training data to test data, and derive theoretical bound of error rate difference between the estimation technique and using perfect reweighing. Zadrozny et al. [42] develop a general

framework for bias correction and apply it into different machine learning models, including bayesian classifiers, logistic regression, SVM and decision tree. Storkey et al. [43] investigate using mixture regression to estimate the weight between the training and test data without presuming the test and training densities are known. Another direction is to use kernel mean matching (KMM) that reweighs training instances to match means of the test ones. Huang et al. [44] first to propose a nonparametric method to correct sample selection bias based on KMM. The authors first use kernel functions to transform the data into feature space, and match training and test distributions in feature space. The advantage is that the resulted optimization is a simple quadratic programming and can be applied straightforwardly into several different regression and classification algorithms. Yu et al. [45] investigate KMM from the theoretical perspective and derive high probability confidence bounds for the KMM, which enhance the understanding of the effectiveness of KMM under covariate shift.

Robust machine learning under distribution shift has been investigated in recent years [46, 47, 48, 49, 50]. In many cases, the density ratio estimation is challenging, e.g. we know nothing more than weak prior knowledge on how the test distribution may shift from the training distribution, or the estimation will incur significant error due to limited labeled data. Wen et al. [47] consider covariate shift between the training and test data and apply Gaussian kernel functions to reweigh the training examples and correct the shift. They propose a minimax robust framework to minimize the most adverse distribution and provide robust classification when it comes to the test data. Similarly, Liu et al. [46] propose a robust bias-aware probabilistic classifier that can deal with different test data distributions using a minimax estimation formulation. Chen et al. [49] borrow the minimax framework formulation from [46] and apply it into the robust regression. The resulted regression model is robust to the uncertainty caused by the sample selection bias and demonstrates the benefits on a

number of regression tasks.

2.3 Fair Machine Learning under Distribution Shift

Most recently, there have been a few papers that study fairness from the distributionally robust perspective. Taskesen et al. [51] propose a distributionally robust logistic regression model with an unfairness penalty. They assume the unknown true test distribution is contained in a Wasserstein ball centered at the empirical distribution on the observed training data. The proposed model, which robustifies the fair logistic regression against all distributions in the ball, is equivalent to a tractable convex problem when unfairness is quantified under the log-probabilistic equalized opportunities criterion. However the approach robustifies the distribution at the individual data level and overlooks the overall distribution. Reza et al. [52] propose the use of ambiguity set to derive the fair classifier based on the principles of distributional robustness. The proposed approach incorporates fairness criteria into a worst case logarithmic loss minimization but ignores the distribution shift. Yurochkin et al. [53] develop an individual fair distributionally robust classifier with a Wasserstein ambiguity set. However, the approach does not admit a tractable convex reformulation.

In addition, there has been research on fairness aware domain adaptation and transfer learning of fairness metrics, e.g., [54, 55, 56, 57]. In particular, Schumann et al. [54] study the fair transfer learning from source domain to target domain and provide a fairness bound on the target domain of the predictor trained on the source domain data. Coston et al. [55] study the fair transfer learning with missing protected attributes under the covariate shift. Similarly, Kallus et al. [57] use covariate shift correction when computing fairness metrics to address bias in label collection. The above works also adopt the idea of reweighing the source examples to resemble the target domain examples during the training.

Moreover, fair learning from label biased training data [58, 59] and fair representation learning [60, 32, 31] are also reported. For example, the authors in [59] develop a framework to model how label bias can arise in a dataset, assuming that there exists an unbiased ground truth, and develop a bias correcting approach based on re-weighting the training examples. Du et al. [61] propose a robust framework to achieve fairness in federated learning under covariate shift between the training data and test data. It applies Gaussian kernel to generate the adversary distribution and minimizes the worst adversary distribution loss during training. Zhao et al. [31] investigate the relationship between fair representation and decision making with theoretical guarantee. They employ the adversarial technique to extract the fair representation. Singh et al. [62] study the covariate shift in the domain adaption and assumes a known causal graph of the data generating process and a context variable causing the shift. Reza et al. [63] apply minimax optimization to achieve robustness fairness under covariate shift, where the fairness violation penalty term between the target input distribution and adversary's conditional label distribution is incorporated during the optimization.

2.4 Distributed and Federated Learning

Distributed deep learning from multiple sources has been long investigated in the past few years. In the pioneer work [64], the dataset is distributed in different machines and a global model is learned by exchanging parameters between participating users. Following this work, later researchers focus on how to train distributed deep learning models more efficiently. For instance, Chilimbi et al. [65] propose an efficient and scalable system to allow the training of distributed deep learning. Wen et al. [66] develop a strategy to reduce communication cost in distributed deep learning to accelerate the training process. More-

over, how to improve the performance of distributed deep learning has been investigated extensively [67, 68, 69]. Distributed deep learning is usually applied in the data center setting, where the data are divided into different powerful machines so that we do not have to concern about the data security, data distribution, or computing power. Federated learning, as a more general distributed learning framework [18], is proposed to push deep learning to mobile and edge devices, in which mobile devices only have limited data and users are willing to collaboratively learn a joint model. Lots of challenges arise for federated learning, including data privacy, non-iid data distribution among different devices, imbalanced data, limited computing power of mobile devices [70, 19].

There exist extensive research works on federated learning aiming to solve various challenges. The pioneer work of federated learning is proposed by [18] where it considers that data is distributed among different mobile devices and proposes an algorithm called FedAvg to enable the collaborative learning among different clients. In this pioneer work, it points out several challenges mainly including the limited bandwidth of federated learning clients, privacy leakage and non-IID (independent and identically distributed) data among different clients. Lots of subsequent works investigate on how to solve the above issues.

The work [18] takes the first shoot to reduce massive communication cost by applying FedAvg to reduce the communication rounds during the model training. The authors in [71] propose to sparsify the exchanged parameters by selecting important ones so that the total number of parameters per round is reduced significantly. In [72], the authors design a gradient quantization approach using less bits to represent the full 32 bits float gradient, which can reduce the communication cost in both uploading and downloading stages. In addition, there are also lots of works toward reducing communication cost [67, 73, 74, 75, 76]. The privacy protection in federated learning also receives increasing attentions in the past few

years. For example, the work [70] encrypts the exchanged parameters of local clients before uploading to the central server. The authors in [77] adopt the concept of differential privacy and add differentially private noise on the exchanged parameters to protect the client privacy. Applying the encryption and differential privacy to prevent privacy leakage are also reported in [78, 79, 80, 81].

The non-IID data widely exists in federated learning due to the massive distributed clients and our work falls into this category. In [19] the authors study the federated learning in a multi-task setting where each client collects individual data with its own statistical pattern. The proposed model learns individual pattern for each client while simultaneously borrow shared information from other clients. The research [82] considers the non-IID distribution among different clients and provides theoretical convergence analysis. The authors in [83] point out the distribution shift in the training data of federated learning and proposes three approaches to adapt the federated learning model to enable the personalization of each local client. In [84] the authors propose a federated learning framework to train the model from non-IID data, where a small subset of data is globally shared among all the clients. Most recently, the authors in [85] propose agnostic federated learning. However, the proposed framework solves the problem by optimizing the worst case for a single client.

3 Preliminaries

In this chapter, we provide the essential notions and fundamental background used in this dissertation. We provide the fairness notions for both classification and regression. Then we provide some basics of the distribution shift.

3.1 Fairness Notions of Classification

We first give the definitions used for fair machine learning classification. Suppose we have a labeled dataset $\mathcal{D}$, consisting of a set of unprotected attributes $\mathbf{X} \in \mathbb{R}^n$, a class label Y and a protected attribute S. For easy discussion, we consider $Y \in \{0, 1\}$ and $S \in \{0, 1\}$ as binary variables. Consider the following classifier $f : \mathbf{X} \to Y$, which maps the unprotected attributes $\mathbf{X}$ to the class label Y. The requirement of fair classification is that the predicted label $\hat{Y} = f(\mathbf{X})$ is unbiased respect to the protected variable S. Previous works [13, 86] provide several definitions to quantify the fairness of the classifier.

Definition 1. Demographic parity. Given a labeled dataset $\mathcal{D}$ and a classifier $f : \mathbf{X} \to Y$, the property of demographic parity is defined as $P(\hat{Y} = 1|S = 1) = P(\hat{Y} = 1|S = 0)$.

In **Definition** 1, $P(\hat{Y} = 1|S = 1)$ denotes the conditional probability of favorable group receiving a positive decision while $P(\hat{Y} = 1|S = 0)$ indicates the conditional probability of unfavorable group receiving a positive decision. We require that the predicted labels are independent of the protected attribute. We usually use risk difference to quantify the unfairness measured by demographic parity.

Definition 2. Risk Difference (RD) is used to measure the discrimination of the model:

$$RD = |\Pr(\hat{Y} = 1|S = 1) - \Pr(\hat{Y} = 1|S = 0)|. \tag{3.1}$$

We can say the model is more fair if RD is smaller. In many fair machine learning frameworks, we usually write down the optimization subject to the fairness constraints as:

$$\begin{aligned} \text{minimize} \quad & f_{\mathcal{D}}(\mathbf{w}) \\ \text{subject to} \quad & RD \leq \tau \end{aligned} \tag{3.2}$$

where $\mathbf{w}$ are the parameters of the machine learning models that need to be optimized and $\tau \in \mathbf{R}^+$ is the threshold of constraint. Usually the optimization of Equation 3.2 under the fairness constraint define by RD is computationally intractable to obtain because the fairness constraint contains the indicator function. An alternative fairness constraint is defined as the covariance between the sensitive attribute and the signed distance from the non-sensitive attribute vector to the decision boundary. It has been proved that the decision boundary fairness is a concept of risk difference [39]. We write this alternative definition $C_{\mathcal{D}}(\mathbf{x}; \mathbf{w})$ as:

$$C_{\mathcal{D}}(\mathbf{x}; \mathbf{w}) = \mathbb{E}[(s - \bar{s})d_{\mathbf{w}}(\mathbf{x})] - \mathbb{E}[(s - \bar{s})]d_{\mathbf{w}}(\mathbf{x}) \propto \sum_{i=1}^{n}(s_i - \bar{s})d_{\mathbf{w}}(\mathbf{x}_i), \tag{3.3}$$

where $\bar{s}$ is the mean value of the protected attribute, $\mathbf{x}_i$ is the ith data sample in $\mathcal{D}$, and s_i is the protected attribute value of $\mathbf{x}_i$. For linear classification models, it has been proved in [**?**] that decision boundary is defined by $\mathbf{x}^T\mathbf{w} = 0$ so that we can reduce Equation 3.3 as

$$C_{\mathcal{D}}(\mathbf{x}; \mathbf{w}) = \sum_{i=1}^{n}(s_i - \bar{s})\mathbf{x}_i^T\mathbf{w} \tag{3.4}$$

3.2 Fairness Notions of Regression

Different from the fair classification, the prediction of the regression classifier is continuous and the protected attribute can either be categorical or numerical. In the following, we use $\hat{y}$ to denote the prediction outcome of regression and a to denote the protected attribute. We first define the used definitions in fairness regression problems.

Definition 3. The mean difference (MD) of numeric prediction $\hat{y}$ in $\mathcal{D}$ by a binary protected attribute a is defined as

$$\mathrm{MD}(\hat{y}, a) = \mathbb{E}(\hat{y}|a = 0) - \mathbb{E}(\hat{y}|a = 1) \tag{3.5}$$

Definition 4. The mean squared error difference (MSED) of numeric prediction $\hat{y}$ in $\mathcal{D}$ by a binary protected attribute a is defined as

$$\mathrm{MSED}(\hat{y}, a) = \mathbb{E}[(y - \hat{y})^2|a = 0] - \mathbb{E}[(y - \hat{y})^2|a = 1] \tag{3.6}$$

Definition 5. The correlation coefficient of numeric prediction $\hat{y}$ and numeric protected attribute a is defined as

$$\rho_{\hat{y}a} = \frac{\mathbb{E}[(\hat{y} - \mu_{\hat{y}})(a - \mu_a)]}{\sigma_{\hat{y}}\sigma_s} \tag{3.7}$$

Definition 6. The partial correlation coefficient of numeric prediction $\hat{y}$ and numeric protected attribute a given y is defined as

$$\rho_{\hat{y}a.y} = \frac{\rho_{\hat{y}a} - \rho_{\hat{y}y}\rho_{ay}}{\sqrt{1 - \rho_{\hat{y}y}^2}\sqrt{1 - \rho_{ay}^2}} \tag{3.8}$$

Definition 7. The statistical parity (SP) is defined as

$$\text{SP} = \mathbb{P}[f(X) \geq z | A = a] - \mathbb{P}[f(X) \geq z] \text{ for all } a \in \mathcal{A} \text{ and } z \in [0, 1] \tag{3.9}$$

Definition 8. The bounded group loss (BGL) is defined as

$$\text{BGL} = \mathbb{E}[l(f(X), Y) | A = a] \text{ for all } a \in \mathcal{A} \tag{3.10}$$

3.3 Distribution Shift

The term of distribution shift in machine learning can also be explained by dataset shift, which was first defined in [8]. It says dataset shift happens if the joint distribution of inputs and outputs differs between the training and test stage. Following [87], we define the dataset shift as:

Definition 9. Dataset shift appears when the training and test joint distributions are different. It is equivalent as

$$P_{tr}(x, y) \neq P_{te}(x, y) \tag{3.11}$$

In **Definition** 9, $P_{tr}(x, y)$ denotes the joint distribution in the training data, while $P_{te}(x, y)$ indicates the joint distribution in the test data. There exists two major types of distribution shift, including covariate shift and prior probability shift.

Definition 10. Covariate shift appears in $\mathbf{X} \to Y$ problem only, and is described as

$$P_{tr}(y|x) = P_{te}(y|x), P_{tr}(x) \neq P_{te}(x) \tag{3.12}$$

Definition 11. Prior probability shift appears in $Y \rightarrow \mathbf{X}$ problem only, and is described as

$$P_{tr}(x|y) = P_{te}(x|y), P_{tr}(y) \neq P_{te}(y) \tag{3.13}$$

Covariate shift is more common in practice and in this dissertation we focus on building robust machine learning models under covariate shift. There exist several possible causes for dataset shift and we focus on sample selection bias in this dissertation, which is the most common one. Sample selection bias usually refers to a systematic flaw in the data collection or labeling process. This flaw leads to the non-uniform sampling of the training data from the population to be modeled. There are four types of sample selection bias analyzed from previous literature [87], including missing completely at random, missing at random, missing at random-class, and missing not at random.

Definition 12. Missing completely at random (MCAR) occurs when the sampling method is completely independent of x and y:

$$P(s = 1|x, y) = P(s = 1) \tag{3.14}$$

Definition 13. Missing at random (MAR) occurs when s depends on x but conditional on x is independent of y:

$$P(s = 1|x, y) = P(s = 1|x) \tag{3.15}$$

Definition 14. Missing at random-class (MARC) occurs when s depends on y but conditional on y is independent of x:

$$P(s = 1|x, y) = P(s = 1|y) \tag{3.16}$$

Definition 15. Missing not at random (MNAR) occurs when there is no independence assumption between x, y, and s.

From **Definitions** 12 - 15, s is a binary selection variable that $s = 1$ ($s = 0$) denotes the inclusion (rejection) of a data in the training sample process. For MCAR, it does not generate any data shift. MAR will cause covariate shift while MARC can lead to prior probability shift. The bias of MNAR will introduce covariate shift or prior probability shift.

4 Fairness-aware Agnostic Federated Learning

4.1 Introduction

In traditional machine learning, the training data is usually stored in a central server. The central server needs to first collect the data from different sources and combines these data together to facilitate the learning. The rapid development of the various machine learning or deep learning models benefits significantly from the large-scale training data. However, the above training approach also raises data privacy concerns because the raw data needs to be uploaded to the server, which can lead to the possible sensitive information leakage.

To alleviate the above issues caused by centralized learning, federated learning is proposed as an attractive framework to handle complex data [18, 19, 20], as it enables a new implementation of distributed deep learning over a large number of clients. Compared to the traditional centralized learning which collects all local data samples and builds the model at a central server, federated learning trains local models on local data samples and local clients exchange parameters to generate a global model. Although tremendous research has been done on the federated learning, how to achieve fairness in federated learning is under-explored. Fairness receives increasing attentions in machine learning. Previous research demonstrates that many machine learning models are often biased or unfair against some protected groups especially when they were trained on biased data. How to achieve fairness in federated learning is more urgent because the training data used in federated learning is often geo-distributed among various groups.

One challenge of achieving fairness in federated learning is due to the statistical challenge of the unknown testing data distribution. In this chapter, we consider a supervised

task with features $\mathbf{X}$ and labels Y and assume $P(y|\mathbf{x})$ is common across all clients. Our goal is to train a single global model that learns $P(y|\mathbf{x})$. The single global model can then be shared to clients or provided to new clients with no training data. However, in federated learning, the distributions of training data at different clients are often different, i.e., $P_i(\mathbf{x}, y) \neq P_j(\mathbf{x}, y)$. Data shift clearly exists between the training distribution of local data used to build the model and the unknown testing distribution. This data shift causes significant challenges for developing fair federated learning algorithms because the learned model may have poor performance on testing data and the learned model with fairness constraint on the training data cannot guarantee the fairness on the testing data.

It is beneficial to build a learning model that is robust against the possible unknown testing distribution in terms of both utility and fairness. The authors in [85] propose agnostic federated learning to deal with the unknown testing data distribution. They model the testing distribution as a mixture of the client data distributions and the mixture weight of one client is deviated from the proportion of its local data in the whole training data. They define the agnostic empirical loss with mixture weights and present a fast stochastic optimization algorithm. However, in their formulation, the model is optimized for the performance of the single worst client and does not take the fairness into consideration.

In this work, we propose fairness-aware agnostic federated learning (AgnosticFair) to deal with the challenge when the testing data distribution is unknown. We formulate AgnosticFair as a two-player adversarial minimax game between the learner and the adversary. The adversary aims to generate any possible unknown testing data distribution to maximize the classifier loss. We assign an individual reweighing value on each training sample and incorporate reweighing value in both agnostic loss function and agnostic fairness constraint.

As a result, the global model learned in the minimax game achieves both high ac-

curacy and fairness guarantee on unknown testing data. Moreover, each client can simply deploy the global model on its local site as the learned global model guarantees fairness on any local data. We conduct extensive experiments on two public datasets and compare our approach with several baselines. Evaluation results demonstrate the effectiveness of our proposed AgnosticFair in terms of accuracy and fairness under data shift scenarios.

The main contributions of this work are summarized as follows:

- To the best of our knowledge, our research is the first one that formulates the problem of fairness-aware federated learning under the data distribution shift among the clients.
- We propose to use kernel function parametrization in loss function and fairness constraints so that they are both agnostic to the data distribution shift.
- We develop an efficient approach to optimize the agnostic loss function under the agnostic fairness constraints between the server and clients. During the optimization process, only parameters and coefficients are needed to exchange between clients and the server without disclosing any raw data.
- We conduct extensive experiments to demonstrate that our approach can achieve fair prediction under distribution shift while maintaining high accuracy.

The rest of this chapter is organized as follows. Section 4.2 presents the problem formulation of fairness agnostic federated learning and techniques to solve the problem. Section 4.3 shows the experimental results of our proposed framework. Section 4.4 concludes the chapter and presents the future work. Note that this chapter is originally from the published work [61].

Table 4.1: Notations

Symbol	Definition
$u_1, u_2, \ldots, u_p$	p clients in federated learning
$\mathcal{D}_k$	local dataset of u_k
$t_i^k = (\mathbf{x}_i^k, y_i^k)$	the i-th tuple of u_k
S	sensitive attribute
$\mathbf{w}$	the parameter vector of federated learning model
$L(\mathbf{w})$	loss function of federated learning model
$L(\mathbf{w}, \boldsymbol{\alpha})$	loss function of agnostic federated learning model
$l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k)$	loss value of t_i^k in u_k
$\boldsymbol{\alpha}$	coefficients of reweighing function
$\theta_{\boldsymbol{\alpha}}(\mathbf{x})$	reweighing function
$K(\mathbf{x})$	Gaussian kernel function
$C_{\mathcal{D}}(\boldsymbol{\alpha}; \mathbf{x}; \mathbf{w})$	agnostic decision boundary fairness constraint
$\mathbf{w}_k^t$	parameters of u_k at tth step
$\bar{\mathbf{w}}^t$	average parameters of p users at tth step
$\boldsymbol{\alpha}^t$	values of $\boldsymbol{\alpha}$ at tth step
L, θ, C	loss, equality, inequality constraint in Equation 4.14
ϕ_L^t, ϕ_C^t	coefficient vector of $\mathbf{w}$ of L, C at tth step
$\phi_{L,k}^t, \phi_{C,k}^t$	coefficient vector of $\mathbf{w}$ of L, C of u_k at tth step
$\psi_L^t, \psi_\theta^t, \psi_C^t$	coefficient vector of $\boldsymbol{\alpha}$ of L, θ, C at tth step
$\psi_{L,k}^t, \psi_{\theta,k}^t, \psi_{C,k}^t$	coefficient vector of $\boldsymbol{\alpha}$ of L, θ, C of u_k at tth step

4.2 Fair Agnostic Federated Learning

4.2.1 Problem Formulation

We first define the following notations used throughout the chapter. Suppose there exist p local clients $u_1, u_2, \ldots, u_p$ in the federated learning setting and each client is associated

with a dataset $\mathcal{D}_k = \{\mathbf{X}, Y\}, k \in [1, p]$. The kth client contains n_k samples and each sample is denoted as $t_i^k : \{\mathbf{x}_i^k, y_i^k\}, i \in [1, n_k]$. The total number of data samples is defined as $n = \sum_{k=1}^{p} n_k$. Let $\mathbf{X} \in \mathcal{X}$ be the input space and $Y \in \mathcal{Y}$ be the output space. We consider the binary classification that $\mathcal{Y} = \{0, 1\}$. The global model f predicts the label as $\hat{y} = f(\mathbf{x})$. The goal of the federated learning is to collaboratively train a machine learning model f by these p clients.

The standard federated learning framework aims to minimize the empirical risk $L(\mathbf{w})$ over all records and learns the parameter vector $\mathbf{w} \in \mathcal{W}$ as the following:

$$\min_{\mathbf{w} \in \mathcal{W}} L(\mathbf{w}) = \frac{1}{n} \sum_{k=1}^{p} \sum_{i=1}^{n_k} l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k) \tag{4.1}$$

where f_k is the classifier of u_k, $l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k)$ is the loss of sample t_i^k in u_k. In general, federated learning includes the following steps.

- Step 1: The server specifies the learning task, e.g., linear regression or deep neural network, for these p clients and sends out initial parameters $\mathbf{w}^0$.
- Step 2: At tth step, u_k receives averaged parameters $\bar{\mathbf{w}}^t$ from the server as the new round of initialization parameters and then finds out the optimal $\mathbf{w}_k^{t+1}$ using its local data $\mathcal{D}_k$:
$$\mathbf{w}_k^{t+1} = \arg\min_{\mathbf{w} \in \mathcal{W}} \frac{1}{n_k} \sum_{i=1}^{n_k} l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k). \tag{4.2}$$
- Step 3: The server receives parameters $\mathbf{w}_k^{t+1}$ from each client u_k and sends averaged parameter $\bar{\mathbf{w}}^{t+1} = \frac{1}{p} \sum_{k=1}^{p} \mathbf{w}_k^{t+1}$ back to all clients.

It should be noted that step 2 and step 3 are repeated until reaching the preset convergence. We present the pseudo code of the standard federated learning framework in Algorithm 1 in

Algorithm 1 Federated Learning Framework

Input:
    $\mathcal{D}_k$ from client u_k, $k = 1, \cdots, p$;
    Training steps T;
    Initial parameter vector $\mathbf{w}^0$;
Output:
    Global model parameter vector $\mathbf{w}$;
Initialize parameters $\mathbf{w}^0$ for all clients;
$t = 0$;
While $t \leq T$ **do**
 Client Side:
 for $k = 1 : p$ **do**
        Client k receives averaged $\bar{\mathbf{w}}^t$ and computes $\mathbf{w}_k^{t+1}$ using Equation 4.2;
        Client k uploads $\mathbf{w}_k^{t+1}$ to server;
 Server Side:
        Server receives $\mathbf{w}_k^{t+1}$ $(1 \leq k \leq p)$ from all clients;
        Server computes $\bar{\mathbf{w}}^{t+1} = \frac{1}{p}\sum_{k=1}^{p} \mathbf{w}_k^{t+1}$ and sends back to each client;
  $t = t + 1$;
return $\bar{\mathbf{w}}^T$

order to compare with our fairness-aware agnostic federated learning in Algorithm 2.

In the fair learning, without loss of generality, we assume S is one sensitive attribute in $\mathbf{X}$ with $S = 0$ representing the minority group and $S = 1$ the majority group. Following the standard federated learning framework, we write the objective function subject to the fairness constraint:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w}) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k} l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k) \\ &\text{subject to} \quad g(\mathbf{x}; \mathbf{w}) \leq \epsilon, \end{aligned} \tag{4.3}$$

where $g(\mathbf{x}; \mathbf{w}) \leq \epsilon$ is the fairness constraint and related to the sensitive attribute S. The weight of each sample in Equation 4.3 from these p clients is uniform. The underlying assumption of the standard federated learning framework is that the testing data distribution is the same as the distribution of the training data (union of data samples from p clients). However, this assumption is rather restrictive and will lead to the following possible draw-

backs. First, the performance of the trained model will be degraded if the distribution of the training data and that of the testing data do not coincide. Second, the fairness achieved on the training data does not guarantee the fairness on the testing data.

4.2.2 Agnostic Loss Function

It is usually considered that the distribution shift exists between the training data $P_{tr}(\mathbf{X})$ and the testing data $P_{te}(\mathbf{X})$, whereas the conditional distribution $P(Y|\mathbf{X})$ indicating the prediction remains the same. To correct the distribution shift between $P_{tr}(\mathbf{X})$ and $P_{te}(\mathbf{X})$, a widely used approach is to reweigh the training samples in the learning process so that the learned model f can reflect the testing data distribution. We write the objective function in Equation 4.3 with the reweighed training samples as the following:

$$\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w}) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta(\mathbf{x}_i^k)l(f_k(\mathbf{x}_i^k;\mathbf{w}), y_i^k), \tag{4.4}$$

where $\theta(\mathbf{x})$ is the reweighing function to correct the distribution shift from $P_{tr}(\mathbf{X})$ to $P_{te}(\mathbf{X})$. There exist several methods to estimate the reweighing value $\theta(\mathbf{x})$ if the (unlabelled) testing data is given [88]. For example, applying density ratio estimation can compute the reweighing value $\theta(\mathbf{x}) = P_{te}(\mathbf{x})/P_{tr}(\mathbf{x})$ for each example, then the reweighing value can represent the possible testing distribution in real scenarios.

However, we cannot properly estimate the reweighing values if we do not have available testing data. Therefore, it is necessary to extend the above framework by building a classifier which is favorable to any unknown testing distribution. We define the agnostic loss

over any unknown testing data as the following:

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\theta\in\Theta} L(\mathbf{w},\theta) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta(\mathbf{x}_i^k)l(f_k(\mathbf{x}_i^k;\mathbf{w}),y_i^k). \tag{4.5}$$

Θ represents the set of unknown testing data distribution produced by the adversary. The formulation can be considered as a two-player adversarial game such that the adversary in Equation 4.5 tries to select a reweighing function $\theta \in \Theta$ to maximize the loss of the objective, whereas the learner tries to find parameters $\mathbf{w} \in \mathcal{W}$ to minimize the worst case loss over the unknown testing data distribution produced by the adversary.

The proposed framework by Equation 4.5 enjoys several advantages. First, under the independent and identically distributed (IID) data settings, the minimization of the robust reweighed loss is equivalent and dual to the empirical risk minimization (objective function in Equation 4.3) [89]. It indicates that the optimization of Equation 4.5 will not cause performance degradation when no distribution shift exists. Second, the optimal $\mathbf{w} \in \mathcal{W}$ is minimized for the worst case loss and the performance of the global model is robust with any unknown testing data.

4.2.3 Kernel Function Parametrization

The reweighing function $\theta \in \Theta$ can be chosen based on the prior knowledge or the application scenario. A reweighing function on individual data sample across clients usually corrects the distribution shift more accurately. We rewrite the agnostic loss in Equation 4.5 as:

$$\begin{aligned}&\min_{\mathbf{w}\in\mathcal{W}} \max_{\boldsymbol{\alpha}\in\mathbf{R}^+} L(\mathbf{w},\boldsymbol{\alpha}) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)l(f_k(\mathbf{x}_i^k;\mathbf{w}),y_i^k)\\ &\text{subject to } \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) = 1.\end{aligned} \tag{4.6}$$

$\theta_{\boldsymbol{\alpha}}(\mathbf{x})$ is the reweighing function that is linearly parametrized as the following:

$$\theta_{\boldsymbol{\alpha}}(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m K_m(\mathbf{x}), 0 \leq \alpha_m \leq B \tag{4.7}$$

where $K_m(\mathbf{x})$ is a basis function, M is the number of basis functions, and $\boldsymbol{\alpha}$ contains the mixing coefficients $\alpha_1, \alpha_2, \cdots, \alpha_M$. The sum-to-one constraint of the reweighing function $\theta_{\boldsymbol{\alpha}}(\mathbf{x})$ ensures that it can properly model the unknown distribution shift from the training data to the testing data. The coefficient α_m is non-negative and bounded by $B \in \mathbf{R}^+$, which constrains the value of $\theta_{\boldsymbol{\alpha}}(\mathbf{x})$ and controls the capacity of the adversary. The linearly parametrized reweighing function $\theta_{\boldsymbol{\alpha}}(\mathbf{x})$ has two advantages. First, the linear form allows us to choose multiple basis functions to capture many different possible uncertainties of the unknown testing data. Second, the optimization with linear form can be more easily solved by linear programming or convex programming tool.

In fact, there are many options to choose the form of basis functions. In this chapter, we choose the Gaussian kernel

$$K_m(\mathbf{x}_i^k) = \exp(- \parallel \mathbf{b}_m - \mathbf{x}_i^k \parallel^2 / 2\sigma^2) \tag{4.8}$$

with the basis $\mathbf{b}_m$ and the kernel width σ. The basis $\mathbf{b}_m$ can be chosen based on some prior knowledge, for example, we can set $\mathbf{b}_m$ as some possible centers of the testing data according to the hypothesis. σ is the width of the kernel. As the smaller variation of $\parallel \mathbf{b}_m - \mathbf{x} \parallel^2$ will cause larger value change of the kernel function and smaller σ indicates more possible testing distributions that the adversary can generate [47].

Or the basis $\mathbf{b}_m$ could be an indicator function $\mathbb{1}_{[.]}$ which represents groups from

different clients, ages, or domains. The value generated by each kernel function can be seen as a conditional probability $P(\mathbf{x}|m)$ of observing $\mathbf{x}$ given the class m in a mixture model. The mixing coefficients $\boldsymbol{\alpha} \in \mathcal{A}$ are usually bounded in the non-negative Euclidean space.

The agnostic federated learning framework proposed by [85] is a special case of our framework. As being said that the basis function could be an indicator function $\mathbb{1}_{[\cdot]}$ representing a group. In [85], it models the testing data distribution as an unknown mixture of p clients where each group is assigned with a uniform weight. The unknown testing data distribution in [85] can be constructed under our framework as the following:

$$\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) = \lambda_k \frac{n_k}{n}, 1 \leq i \leq n_k, 1 \leq k \leq p \tag{4.9}$$

where $\frac{n_k}{n}$ is the uniform weight of each data $\mathbf{x}_i^k$ before reweighing. More specifically, for each client u_k, the agnostic federated learning [85] assigns the same value λ_k to reweigh each data in $\mathcal{D}_k$. However, assigning reweighing value at the client level is insufficient to model the unknown distribution shift due to the following two reasons. First, the data from the same client also has diversity and needs different reweighing values. Second, different clients can have similar data and these similar data should be assigned with similar reweighing values. In our framework, we assign the reweighing value at the individual data level, which is more capable of modeling the unknown testing data distribution.

4.2.4 Agnostic Fairness Constraint

The fairness constraint $g(\mathbf{x}; \mathbf{w})$ in standard federated learning (Equation 4.3) assigns uniform weight for each sample. When it comes to the unknown testing data, the fairness achieved by Equation 4.3 may not guarantee the fairness on unknown testing data. As

being said, the adversary tries to produce a set of possible unknown testing distributions. It encourages us to construct the agnostic fairness constraint based on the unknown testing distributions by the adversary. Then, the optimization of the objective function is subject to the fairness constraint based on the unknown testing distribution.

There exist several notions for fairness constraint and we have described the definitions in Chapter 3. Here we use the risk difference RD as the fairness constraint. For each client, we define $\mathcal{D}_{ij}^k = \{\mathbf{x}|\hat{Y} = i, S = j\}$ where $i, j \in [0, 1]$. For notation convenience, we define $\mathcal{D}_{\cdot j}^k = \{\mathbf{x}|S = j\}$ where $j \in [0, 1]$ and $\cdot$ represents $\{0, 1\}$. Then we can write the expression for $RD(f)$ with uniform weight on training data as the following:

$$|\frac{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k \in \mathcal{D}_{11}^k}}{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k \in \mathcal{D}_{\cdot 1}^k}} - \frac{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k \in \mathcal{D}_{10}^k}}{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k \in \mathcal{D}_{\cdot 0}^k}}| \leq \epsilon, \tag{4.10}$$

where $\mathbb{1}_{[.]}$ is an indicator function and $\epsilon \in [0, 1]$ is a threshold for the fairness constraint. However, Equation 4.10 is constructed based on the training data and cannot preserve fairness on the unknown testing data. Hence, we use the same reweighing function to construct the agnostic fairness constraint as the following:

$$|\frac{\sum_{k=1}^{p}\sum_{\mathbf{x}_i^k \in \mathcal{D}_{11}^k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)}{\sum_{k=1}^{p}\sum_{\mathbf{x}_i^k \in \mathcal{D}_{\cdot 1}^k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)} - \frac{\sum_{k=1}^{p}\sum_{\mathbf{x}_i^k \in \mathcal{D}_{10}^k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)}{\sum_{k=1}^{p}\sum_{\mathbf{x}_i^k \in \mathcal{D}_{\cdot 0}^k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)}| \leq \epsilon. \tag{4.11}$$

Then our proposed fairness-aware agnostic federated learning (AgnosticFair) is the combination of Equation 4.6 and Equation 4.11. The fairness constraint in Equation 4.11 is constructed based on the unknown testing distribution, so when it comes to the unknown testing data, the trained classifier can still preserve the fairness. Another benefit is that even though the distributions of the local clients and the server side do not coincide, the classifier

can still guarantee the fairness on each local client due to the agnostic fairness constraint.

The optimal solution of Equation 4.6 under the fairness constraint by Equation 4.11 is computationally intractable to obtain because the fairness constraint contains the indicator function. We then can use the covariance fairness (Equation 3.3) $C_{\mathcal{D}}(\mathbf{x}; \mathbf{w})$ as:

$$C_{\mathcal{D}}(\mathbf{x}; \mathbf{w}) = \frac{1}{n} \sum_{k=1}^{p} \sum_{i=1}^{n_k} (s_{\mathbf{x}_i^k} - \bar{s}) d_{\mathbf{w}(\mathbf{x}_i^k)}, \tag{4.12}$$

where $s_{\mathbf{x}_i^k}$ is the value of the sensitive attribute of the sample $\mathbf{x}_i^k$, $d_{\mathbf{w}(\mathbf{x}_i^k)}$ is the distance to the decision boundary of the classifier f, $\bar{s}$ is the mean value of the sensitive attribute over $\mathcal{D}$ that is $\frac{\sum_{k=1}^{p} \sum_{i=1}^{n_k} s_{\mathbf{x}_i^k}}{n}$. To achieve fair classification, it is required that $|C_{\mathcal{D}}(\mathbf{x}; \mathbf{w})| \leq \tau$ where $\tau \in \mathbf{R}^+$. Incorporating the reweighing values into the fairness constraint gives:

$$C_{\mathcal{D}}(\boldsymbol{\alpha}; \mathbf{x}; \mathbf{w}) = \frac{1}{n} \sum_{k=1}^{p} \sum_{i=1}^{n_k} (s_{\mathbf{x}_i^k} - \bar{s}) \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) d_{\mathbf{w}(\mathbf{x}_i^k)}. \tag{4.13}$$

4.2.5 Solving Fairness-aware Agnostic Federated Fairness Learning

Now we are ready to formulate our agnostic federated learning under the decision boundary fairness constraint as:

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W}} \max_{\boldsymbol{\alpha} \in \mathbf{R}^+} \quad & L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{n} \sum_{k=1}^{p} \sum_{i=1}^{n_k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k) \\ \text{subject to} \quad & \frac{1}{n} \sum_{k=1}^{p} \sum_{i=1}^{n_k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) = 1, 0 \leq \alpha_m \leq B \\ & |\frac{1}{n} \sum_{k=1}^{p} \sum_{i=1}^{n_k} (s_{\mathbf{x}_i^k} - \bar{s}) \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) d_{\mathbf{w}(\mathbf{x}_i^k)}| \leq \tau. \end{aligned} \tag{4.14}$$

The optimization of Equation 4.14 includes two sets of parameters, $\boldsymbol{\alpha}$ and $\mathbf{w}$. The minimax expression encourages us to alternatively optimize $\boldsymbol{\alpha}$ and $\mathbf{w}$ in an iterative way.

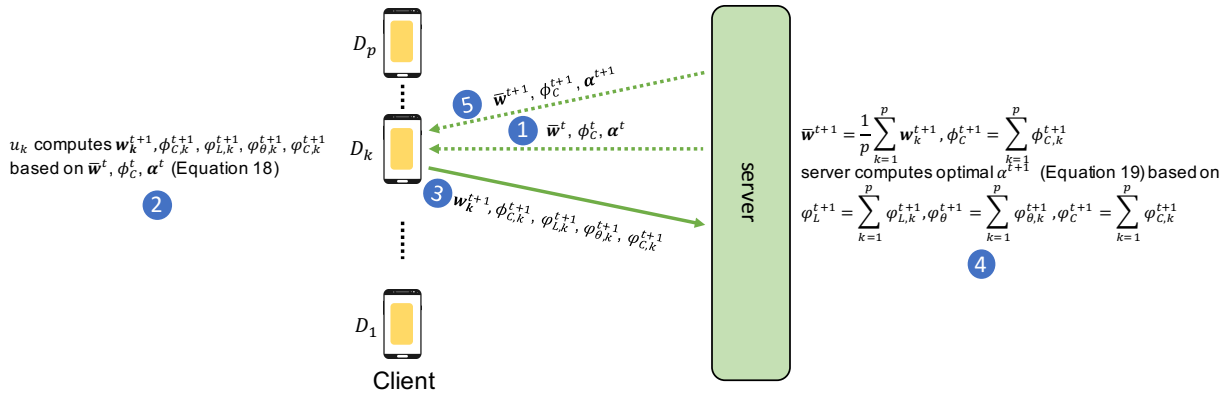

Figure 4.1: The interaction between the client and server in federated learning. The client side optimizes its local $\mathbf{w}$ and server side optimizes the $\boldsymbol{\alpha}$.

The client and server will collaboratively optimize $\mathbf{w}$ and $\boldsymbol{\alpha}$ to solve the minimax problem. The general pipeline is that the client optimizes $\mathbf{w}$ with fixed $\boldsymbol{\alpha}$, while the server optimizes $\boldsymbol{\alpha}$ with fixed $\mathbf{w}$. One challenge is how both the server and the clients conduct optimization iteratively through sharing parameters or intermediate results (rather than raw data), as required in federated learning.

The objective loss $L(\mathbf{w}, \boldsymbol{\alpha})$ (abbreviated as L) shown in Equation 4.14 can be written as a function of $\boldsymbol{w}$ with corresponding coefficients ϕ_L when $\boldsymbol{\alpha}$ is fixed. More importantly, the second summation over samples in client u_k, $\sum_{i=1}^{n_k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) l(f_k(\mathbf{x}_i^k; \mathbf{w}), y_i^k)$, can be similarly expressed as a function $\boldsymbol{w}$ with corresponding coefficients $\phi_{L,k}$. We can easily see $\phi_L = \sum_{k=1}^{p} \phi_{L,k}$ and hence each client can simply send $\phi_{L,k}$ (rather than any raw data) to the server. Similarly, when $\boldsymbol{w}$ is fixed, $L(\mathbf{w}, \boldsymbol{\alpha})$ can be written as a function of $\boldsymbol{\alpha}$ with corresponding coefficients ψ_L, the second summation over samples in client u_k has coefficients $\psi_{L,k}$, and $\psi_L = \sum_{k=1}^{p} \psi_{L,k}$.

Similarly, the equality constraint for the reweighing functions (abbreviated as θ) and the inequality constraint for the decision boundary fairness (abbreviated as C) in Equation

4.14 can be expressed as functions with corresponding coefficients. Throughout this chapter, we use ϕ to denote the coefficients vector of $\mathbf{w}$ (with fixed α), ψ to denote the coefficients vector of α (with fixed $\mathbf{w}$). We use the subscript L, θ and C denote the loss function, equality constraint, and inequality constraint, and further add the subscript k for the coefficients from client u_k. Moreover, we use the superscript t to express the coefficients at step t during the optimization. For example, $\psi_{C,k}^t$ denotes the coefficient vector of α (with fixed $\mathbf{w}$) in the inequality constraint formula for client u_k at step t. We note that the equality constraint only involves variable $\boldsymbol{\alpha}$ and does not have coefficient vector ϕ_θ. We put all of the notations in Table 4.1 and show their relationships in the following equation.

$$\begin{aligned} \phi_L^t &= \sum_{k=1}^{p} \phi_{L,k}^t, \phi_C^t = \sum_{k=1}^{p} \phi_{C,k}^t \\ \psi_L^t &= \sum_{k=1}^{p} \psi_{L,k}^t, \psi_\theta^t = \sum_{k=1}^{p} \psi_{\theta,k}^t, \psi_C^t = \sum_{k=1}^{p} \psi_{C,k}^t. \end{aligned} \tag{4.15}$$

In the following, we present details about the optimization process between the server and client. We show those key parameters and coefficients exchanged between the client and the server as well as their calculations in Figure 4.1.

Client side: In standard federated learning, each client computes $\mathbf{w}$ based on its local data and exchanges the updated $\mathbf{w}$ with other clients via the server. Here we also follow this standard approach that each client computes the optimal values of $\mathbf{w}$ locally using the fixed $\boldsymbol{\alpha}$ received from the server. We can decompose the part of Equation 4.14 related to $\mathbf{w}$ as the following:

$$\begin{aligned} \min_{\mathbf{w}\in\mathcal{W}} \quad & L(\mathbf{w}) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k} \theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) l(f_k(\mathbf{x}_i^k;\mathbf{w}), y_i^k) \\ \text{subject to} \quad & |\frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k} (s_{\mathbf{x}_i^k} - \bar{s})\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) d_{\mathbf{w}(\mathbf{x}_i^k)}| \leq \tau. \end{aligned} \tag{4.16}$$

Algorithm 2 AgnosticFair: Fairness-aware Agnostic Federated Learning

Input:
    $\mathcal{D}_k$ from client u_k, $k = 1, \cdots, p$;
    Training steps T;
    Initial parameters $\mathbf{w}^0$ and $\boldsymbol{\alpha}^0$;
Output:
    Global model parameter vector $\mathbf{w}$;
Initialize parameters $\mathbf{w}^0$ and $\boldsymbol{\alpha}^0$ for all clients;
$t = 0$;
While $t \leq T$ **do**
 Client Side:
 for $k = 1 : p$ **do**
        Client k receives averaged $\bar{\mathbf{w}}^t$, $\boldsymbol{\alpha}^t$ and ϕ_C^t;
        Client k computes optimal $\mathbf{w}_k^{t+1}$ using Equation 4.17 and uploads to server;
        Client k computes $\phi_{C,k}^t, \psi_{L,k}^{t+1}, \psi_{\theta,k}^{t+1}, \psi_{C,k}^{t+1}$ and uploads to server;
 Server Side:
        Server aggregates $\psi_L^{t+1} = \sum_{k=1}^p \psi_{L,k}^{t+1}$, $\psi_\theta^{t+1} = \sum_{k=1}^p \psi_{\theta,k}^{t+1}, \psi_C^{t+1} = \sum_{k=1}^p \psi_{C,k}^{t+1}$;
        Server computes optimal $\boldsymbol{\alpha}^{t+1}$ using Equation 4.18;
        Server aggregates $\phi_C^{t+1} = \sum_{k=1}^p \phi_{C,k}^{t+1}$ and averages $\bar{\mathbf{w}}^{t+1} = \frac{1}{p}\sum_{k=1}^p \mathbf{w}_k^{t+1}$;
        Server sends back $\bar{\mathbf{w}}^{t+1}$, $\boldsymbol{\alpha}^{t+1}$ and ϕ_C^{t+1};
  $t = t + 1$;
return $\bar{\mathbf{w}}^T$

It can be seen that given the fixed $\boldsymbol{\alpha}$, the optimization of $\mathbf{w}$ is only subject to the inequality constraint. The optimization of Equation 4.16 depends on the choice of loss function and the learning model. For example, the loss function of linear regression is convex and the inequality constraint of $\mathbf{w}$ is also linear so the convex programming tool can be used to solve it. However, the loss function over $\mathbf{w}$ of many other machine learning models (e.g., deep learning models) is not convex. Hence, it is challenging to optimize the non-convex function subject to the constraint. We observe that the inequality constraint in Equation 4.16 is used to guarantee the fairness of the updated $\mathbf{w}$ during the optimization. Instead of optimizing non-convex loss function subject to the constraints, we can transform the inequality constraint to a penalty term on the loss function.

We choose the square term for fairness inequality constraint as a penalty and rewrite

the loss function of $\mathbf{w}$ for client k as the following:

$$\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w}) = \frac{1}{n_k}\sum_{i=1}^{n_k}[\theta(\mathbf{x}_i^k)l(f_k(\mathbf{x}_i^k;\mathbf{w}),y_i^k)] + \lambda(\frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}(s_{\mathbf{x}_i^k}-\bar{s})\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)d_{\mathbf{w}(\mathbf{x}_i^k)}-\tau)^2 \tag{4.17}$$

where λ is a hyperparameter controlling the trade-off between the classification accuracy and the fairness. Equation 4.17 includes two terms. The first term is the loss of each client based on its own data while the second term is the global fairness constraint.

Suppose client u_k receives the average parameters $\bar{\mathbf{w}}^t$ and $\boldsymbol{\alpha}^t$ from the server at the tth step, it can compute $\phi_{L,k}^t$ using $\boldsymbol{\alpha}^t$ and local data $\mathcal{D}_k$. For the second term computation, it needs to receive $\phi_C^t = \sum_{k=1}^p \phi_{C,k}^t$ from the server, where each client can compute $\phi_{C,k}^t$ independently using local data $\mathcal{D}_k$. In this process, the raw data of each client is not exposed, which fulfills the requirement of federated learning. Given $\bar{\mathbf{w}}^t$, $\phi_{L,k}^t$ and ϕ_C^t, client u_k obtains the complete form of Equation 4.17 and can compute optimal $\mathbf{w}_k^{t+1}$ based on $\mathcal{D}_k$. Based on $\mathbf{w}_k^{t+1}$ and fixed $\boldsymbol{\alpha}^t$, it can compute $\psi_{L,k}^{t+1}, \psi_{\theta,k}^{t+1}, \psi_{C,k}^{t+1}$, and $\phi_{C,k}^{t+1}$ and upload them to the server.

Server side: The optimization of $\boldsymbol{\alpha}$ is subject to both equality and inequality constraints, which is expressed as:

$$\begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbf{R}^+} \quad & L(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)l(f_k(\mathbf{x}_i^k;\mathbf{w}),y_i^k) \\ \text{subject to} \quad & \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) = 1, 0 \le \alpha_m \le B \\ & |\frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}(s_{\mathbf{x}_i^k}-\bar{s})\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)d_{\mathbf{w}(\mathbf{x}_i^k)}| \le \tau. \end{aligned} \tag{4.18}$$

Given fixed $\mathbf{w}$, $\theta_{\boldsymbol{\alpha}}$ is a linear function subject to linear equality and inequality constraints. The server aggregates coefficient vector $\psi_L^{t+1}, \psi_\theta^{t+1}, \psi_C^{t+1}$ of $\boldsymbol{\alpha}$ (Equation 4.15) to obtain the

complete form of Equation 4.18.

The server then uses linear programming tool to obtain the optimal values $\boldsymbol{\alpha}^{t+1}$ and sends back to each client. In addition, the server also averages parameters $\bar{\mathbf{w}}^{t+1} = \frac{1}{p}\sum_{k=1}^{p}\mathbf{w}_k^{t+1}$, aggregates $\phi_C^{t+1} = \sum_{k=1}^{p}\phi_{C,k}^{t+1}$, and sends them back to each client for next round iteration.

We also present the pseudo code of our proposed fairness-aware agnostic federated learning (AgnosticFair) in Algorithm 2. It can be seen that each client optimizes $\mathbf{w}$ at the local side and the server optimizes $\boldsymbol{\alpha}$. The final classifier with fair prediction is achieved through the iterative optimization process.

4.2.6 Variants of AgnosticFair

There exist several variants of our fairness-aware agnostic federated learning which can be used as baselines. We consider the following two variations in this chapter and will show their experimental results in Section 4.3.

The first variation is termed as AgnosticFair-a that optimizes agnostic loss (Equation 4.6) without any fairness constraint. AgnosticFair-a only deals with the data distribution shift regarding the accuracy and has no fairness guarantee on the model. The goal of considering AgnosticFair-a is to test its accuracy performance with comparison to [85] using our proposed reweighing function.

The second variation is termed as AgnosticFair-b that considers the agnostic loss and

uniform weighted fairness constraint, which is expressed as the following:

$$\begin{aligned} \min_{\mathbf{w}\in\mathcal{W}} \max_{\boldsymbol{\alpha}\in\mathbf{R}^+} \quad & L(\mathbf{w},\boldsymbol{\alpha}) = \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k)l(f_k(\mathbf{x}_i^k;\mathbf{w}),y_i^k) \\ \text{subject to} \quad & \frac{1}{n}\sum_{k=1}^{p}\sum_{i=1}^{n_k}\theta_{\boldsymbol{\alpha}}(\mathbf{x}_i^k) = 1, 0 \le \alpha_m \le B \\ & |\frac{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k\in\mathcal{D}_{11}^k}}{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k\in\mathcal{D}_{\cdot 1}^k}} - \frac{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k\in\mathcal{D}_{10}^k}}{\sum_{k=1}^{p}\sum \mathbb{1}_{\mathbf{x}_i^k\in\mathcal{D}_{\cdot 0}^k}}| \le \epsilon. \end{aligned} \tag{4.19}$$

It is expected that AgnosticFair-b can guarantee the fairness under the IID setting but fails to achieve fairness under unknown distribution shift. For comparison, our AgnosticFair (Equation 4.14) can guarantee fairness while maintain high accuracy performance under the unknown distribution shift. We will show their comparisons in Section 4.3.

4.3 Experiments

4.3.1 Experimental Setup

Datasets. We evaluate our proposed approach AgnosticFair on two datasets, Adult dataset [90] and Dutch dataset [28]. Adult dataset collects the personal information from different people including age, education level, race, gender, and so forth. The prediction task is to determine whether the income of a person is over 50K or not. Dutch dataset collects personal information of the inhabitants in Netherlands and the task is also to classify the individual into high income or low income. For both datasets, we set "gender" as the sensitive attribute. For non-sensitive attributes, we apply one-hot encoding to convert the categorical attributes into vectors and normalize numerical attributes to the range within $[0, 1]$. After preprocessing, Adult dataset consists of 45222 data samples and each data sample has 40 features, whereas Dutch dataset consists of 60420 data samples and each data sample has

Table 4.2: Model performance under data distribution shift (Adult and Dutch) Acc: accuracy

Methods	**Adult Dataset**			**Dutch Dataset**		
	Training Acc	Testing Acc	Testing RD	Training Acc	Testing Acc	Testing RD
FL	0.7500	0.7998	0.1477	0.8133	0.7951	0.1945
AgnosticFair	0.7413	0.7626	0.0196	0.7478	0.7205	0.0371
AgnosticFair-a	0.7820	0.8294	0.1306	0.8259	0.8162	0.2154
FairFL	0.7537	0.7534	0.0852	0.6899	0.7011	0.0961
[85]	0.7761	0.7774	0.1150	0.8170	0.7738	0.1238

Table 4.3: Local fairness and global fairness under data distribution shift (Adult)

Methods	u_1 Testing RD	u_2 Testing RD	Global Testing RD	Global Testing Accuracy
AgnosticFair	0.0208	0.0177	0.0196	0.7626
AgnosticFair-b	0.0450	0.0795	0.0673	0.7885

35 features.

To create the distribution shift scenarios from the training set to the testing set, we artificially split each dataset as the following. For Adult, the training set $\mathcal{D}^{tr}$ contains 80% data of people working in private company and 20% data of people working in other organizations, and the testing set $\mathcal{D}^{te}$ contains the rest of the data. Hence, $\mathcal{D}^{tr}$ of Adult is dominated by data of people working in private company, while $\mathcal{D}^{te}$ of Adult is dominated by data of people working in other organizations. We consider 2 local clients in our experiment u_1 and u_2. u_1 only contains data of people working in private company from $\mathcal{D}^{tr}$ while u_2 only contains data of people working in other groups. Similarly for Dutch, the training set $\mathcal{D}^{tr}$ contains 80% data of people who are married with children and 40% data of people from other groups, and $\mathcal{D}^{te}$ contains the rest of the data. We also consider 2 local clients, u_1 contains data of people who are married with children while u_2 contains of people in other groups.

Hyperparameters. In our experiment, we use Gaussian kernel in Equation 4.8 as the

Table 4.4: Model performance of IID data(Adult and Dutch) Acc: accuracy

Methods	Adult Dataset			Dutch Dataset		
	Training Acc	Testing Acc	Testing *RD*	Training Acc	Testing Acc	Testing *RD*
FL	0.8129	0.8130	0.1490	0.8116	0.8096	0.1698
AgnosticFair	0.7938	0.7749	0.0299	0.7338	0.7322	0.0270
AgnosticFair-a	0.8083	0.8111	0.1515	0.8135	0.8089	0.1526
FairFL	0.7731	0.7723	0.0235	0.7564	0.7346	0.0325
[85]	0.7774	0.7785	0.1484	0.7925	0.7892	0.1547

reweighing function to construct the unknown testing data distribution. The upper bound B for $\boldsymbol{\alpha}$ is set as 5 and σ is chosen to be 1. In fact, the upper bound of B is rarely reached in practical optimization so that it will not limit the power of the adversary too much. The basis of the Gaussian kernel is chosen from training data and the number of kernels is set as 200. The threshold τ in Equation 4.14 is set as a constant 0.05 and λ in Equation 4.17 is set as 2.

Baselines. In our experiment, we use the logistic regression model to evaluate our proposed algorithm. We compare the performance of our proposed AgnosticFair with the following baselines: (a) standard federated learning (FL) without fairness constraint; (b) standard federated learning with fairness constraint (FairFL); (c) agnostic federated learning [85] that assigns the reweighing value at the client level. To conduct meaningful comparison between our model with baselines, we introduce several variations: AgnosticFair-a that optimizes agnostic loss (Equation 4.6) without any fairness constraint; AgnosticFair-b that optimizes agnostic loss subject to unweighted fairness constraint (Equation 4.19).

Metrics. We evaluate our proposed framework and baselines based on utility and fairness. We use accuracy to measure the utility and risk difference (RD) to measure the fairness. A fair classifier usually has $RD(f) \leq 0.05$. We run all experiments 20 times and report the average results.

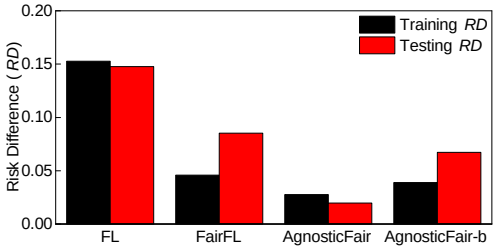

Figure 4.2: Model fairness under data distribution shift (Adult)

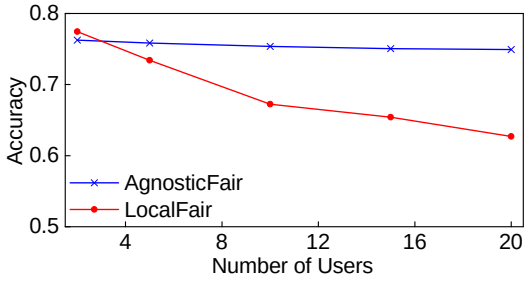

Figure 4.3: The accuracy of the global model with different number of local clients (Adult).

4.3.2 Comparison under Unknown Data Shift

In our framework, the reweighing function is designed to improve the performance of the classifier if the data distributions of the training and the testing set do not coincide. The use of the reweighing function in the fair constraint can also achieve fairness guarantee under unknown testing data.

Accuracy. We report the experimental results in Table 4.2 that demonstrate the accuracy improvement by our framework. We summarize several interesting points regarding accuracy as the following. The testing accuracy of AgnosticFair-a is 0.8264 on Adult and 0.8162 on Dutch, while the accuracy of FL is 0.7998 on Adult and 0.7951 on Dutch. More specifically, AgnosticFair-a outperforms FL by 0.0296 on Adult and by 0.0211 on Dutch. It demonstrates that the agnostic loss function in Equation 4.5 improves the performance of the model when it comes to the distribution change.

Compared to [85], AgnosticFair-a enjoys higher accuracy. As we discussed before,

assigning a reweighing value at the client level optimizes the client with the worst loss which will reduce its generalization ability on the unknown testing data. For AgnosticFair-a, we assign a reweighing value for each data sample across all clients and put more weights on difficult data samples with higher loss. In fact, the difficult data samples can come from any client and are taken into the consideration during the training process. Hence, the generalization ability in the testing stage of AgnosticFair-a is increased.

Fairness. In this experiment, we will show that our proposed AgnosticFair can achieve fairness guarantee under unknown data distribution shift. In Table 5.1, the two columns "Testing RD" show the risk difference values of our Agnostic-Fair and four baselines over the testing data of both Adult and Dutch. We also draw a plot in Figure 4.2 to show achieving fairness on the training data by baselines cannot guarantee fairness on testing data whereas our AgnosticFair can achieve the guarantee. First, experimental results show that both AgnosticFair and AgnosticFair-b can achieve fairness on the training data, but only AgnosticFair can guarantee the fairness on the testing data. AgnosticRegFair-b uses the agnostic loss function, but its fairness constraint is unweighted. It can be concluded that using the agnostic loss function only cannot guarantee the fairness when it comes to the unknown testing data.

Second, FL achieves high accuracy but cannot achieve the fairness. The RD of the FL is 0.1477, whereas a fair learning model usually requires RD to be less than 0.05. The fairness constraint of FairFL does not consider data distribution shift. The results show that FairFL achieves fairness on the training data but fails on the testing data.

Federated Learning with Different Number of Clients. Our proposed AgnosticFair can also achieve fairness for local clients when the distribution shift exists between the local clients and the global server. In fact, the distribution shift from the global training data

to the local client data is a special case of the unknown testing data distribution. We use the same data split setting and report the result of local fairness and global fairness in Table 4.3. It can be seen that AgnosticFair-b cannot guarantee the fairness on the unknown testing data because it fails on u_2 ($RD = 0.0795$). Due to the agnostic fairness constraint of AgnosticFair, it can achieve fairness for both local clients. To demonstrate the stability of our proposed AgnosticFair, we show its accuracy under different number of clients in Figure 4.3. We use the same data split setting and distribute the data evenly to each client without overlap. The accuracy of the global classifier is recorded when fairness is achieved on all clients. It can be seen that the performance of AgnosticFair is independent of the number of clients. For comparison, we also use another straightforward approach LocalFair that achieves fairness for each local client by adding a local fairness constraint based on its own data. It can guarantee the fairness for local client, however, the drawback is to add a local fairness constraint for each client, which will reduce the utility of the global model if more clients are included. Figure 4.3 also shows the accuracy curve of LocalFair, we can see that its performance degrades significantly with the increasing number of clients.

4.3.3 Comparison under IID Setting

In our last experiment, we also test the performance of our model under the IID data setting. In this experiment, we randomly split two datasets, Adult and Dutch. For each dataset, we use 80% of the data as the training set and the rest 20% as the testing set. The number of local clients is set to be 2 and the training data is evenly distributed to each local client. Table 4.4 shows the experimental results.

First, AgnosticFair-a achieves same level of performance with FL if no data distribution shift exists. For Adult, the testing accuracy of FL (AgnosticFair-a) is 0.8130 (0.8111).

For Dutch, the testing accuracy of FL (AgnosticFair-a) is 0.8096 (0.8089). It is quite interesting because in [89] the authors state: under (IID) data, the minimization of the robust reweighed loss (Equation 4.4 in our framework) is equivalent and dual to the empirical risk minimization (Equation 4.3 in our framework). Hence, the accuracy of the AgnosticFair-a and FL also echoes the theoretical statement in [89].

Second, our proposed AgnosticFair-a also achieves higher accuracy than [85] under the IID setting. More specially, the testing accuracy is 0.7785 (0.7892) for Adult (Dutch), which is still lower than that of FL. As aforementioned, [85] assigns a different weight for each client. The optimization process as stated in their work will improve the worst loss of the individual client, whereas the global generalization on the testing data will be weakened. Finally, our AgnosticFair achieves fairness guarantees while preserve good accuracy.

4.4 Summary

In this chapter, we develop a fairness-aware agnostic federated learning framework (AgnosticFair) to deal with the challenge of unknown testing distribution. We use kernel reweighing functions to assign a reweighing value on each training sample in both loss function and fairness constraint. Therefore, the centralized model built from AgnosticFair can achieve high accuracy and fairness guarantee on unknown testing data. Moreover, the built model can be directly applied to local sites as it guarantees fairness on local data distributions. Experimental results on two real datasets demonstrate the effectiveness in terms of both utility and fairness under data shift scenarios.

5 Robust Fairness-aware Learning under Sample Selection Bias

5.1 Introduction

Traditional supervised machine learning assumes that training data and test data are independently and identically distributed (iid), i.e., each example t with pairs of feature input x and label y drawn from the same distribution $\mathcal{Q} = P(x, y)$. The conditional label distribution, $P(y|x)$, is estimated as $\hat{P}(y|x)$ (aka, a classifier $f(x)$) from the given training dataset $\mathcal{D}_s$. Similarly, in the fair machine learning, we aim to learn a fair classifier $f(x, a)$ from the training dataset drawn from $\mathcal{Q} = P(x, a, y)$ where a is a protected attribute such as gender or race. However, when the distributions on training and test data sets do not match, we are facing sample selection bias or covariate shift. Sample selection bias occurs frequently in reality — the available training data have been collected in a biased manner whereas the test is instead performed over a more general population. The classifier f simply learned from the training dataset is vulnerable to sample selection bias and will incur more accuracy loss over test data. Moreover, the fair classifier trained with the biased data cannot guarantee fairness over test data. This is a serious concern when it is critical and imperative to achieve fairness in many applications.

In this chapter, we develop a framework for robust and fair learning under sample selection bias. We embrace the uncertainty incurred by sample selection bias by producing predictions that are both fair and robust in test data. Our framework adopts the reweighing estimation approach for bias correction and the minimax robust estimation approach for achieving robustness on prediction accuracy. Moreover, during the minimax optimization, the fairness is achieved under the worst case, which guarantees the model's fairness on test

data. To address the intractable issue, we approximate the fairness constraint using the boundary fairness and combine into the classifier's loss function as a penalty. The modified loss function is minimized in view of the most adverse distribution within a Wasserterin ball centered at the empirical distribution of the training data.

We present two algorithms, *RFLearn*^{1} for the scenario where the unlabeled test dataset $\mathcal{D}$ is available, and *RFLearn*^{2} for the scenario where $\mathcal{D}$ is unavailable. In *RFLearn*^{1}, we estimate the sample selection probability via its density ratio of training data and test data and then correct the bias in loss function. In *RFLearn*^{2}, we introduce some natural assumptions, i.e., the samples in the same cluster have the same selection probability which is within a range from the uniform selection probability. The algorithm first clusters the training data and robustifies the sample selection probability estimation of each cluster within a Wasserterin ball. We test our algorithms on two real-world datasets and experimental results demonstrate that our algorithms can achieve both good performance on prediction and fairness. Note that this chapter is originally from the published work [91].

5.2 Problem Formulation

We first define notations throughout this chapter. Let X denote the feature space, A the protected attribute, and Y the label set. Let $\mathcal{Q}$ denote the true distribution over $X \times A \times Y$ according to which test samples $t = (x, a, y)$ are drawn.

For simplicity, we assume both y and a are binary where $y = 1$ (0) denotes the favorable (unfavorable) decision and $a = 1$ (0) denotes the majority (minority) group. Under the sample selection bias setting, the learning algorithm receives a training dataset $\mathcal{D}_s$ of $N_{\mathcal{D}_s}$ labeled points $t_1, \cdots, t_{N_{\mathcal{D}_s}}$ drawn according to a biased distribution $\mathcal{Q}_s$ over $X \times A \times Y$. This sample bias can be represented by a random binary variable s that controls the selection

of points, i.e., $s = 1$ for selected and $s = 0$ otherwise. We consider the problem of building over $\mathcal{D}_s$ a fair classifier f assigning labels $\hat{Y} \in \{0, 1\}$ that depends only on X, A (so $\hat{Y} \perp (Y, S)|X, A$) under certain fairness constraints. Many fairness notions were proposed in the literature, such as demographic parity and mistreatment parity. In this chapter, we choose the classic demographic parity and use risk difference (RD) as the fairness quantity. In short, RD measures the difference of the positive predictions between the majority group and minority group.

Problem Formulation 1 (Fair Classifier Under Sample Selection Bias) With the observed $\mathcal{D}_s$, how to construct a fair classifier f that minimizes the expected loss $\mathbb{E}_{(x,a,y)\in\mathcal{Q}}[l(f(x, a), y)]$ subject to $|RD(\mathcal{Q})| \leq \tau$ where l is the loss function, $RD(\mathcal{Q})$ is the risk difference over distribution $\mathcal{Q}$, i.e., $RD(\mathcal{Q}) = |P_{\mathcal{Q}}(\hat{y} = 1|a = 1) - P_{\mathcal{Q}}(\hat{y} = 1|a = 0)|$, and $\tau \in [0, 1]$ is a threshold for the fairness constraint.

5.3 Fair Classifier under Sample selection bias

The probability of drawing $t = (x, a, y)$ according to the true but unobserved distribution $\mathcal{Q}$ is straightforwardly related to the observed distribution $\mathcal{Q}_s$. By definition of the random selection variable s, the observed biased distribution $\mathcal{Q}_s$ can be expressed by $P_{\mathcal{Q}_s}(t) = P_{\mathcal{Q}}(t|s = 1)$ or $P_{\mathcal{Q}_s}(x, a, y) = P_{\mathcal{Q}}(x, a, y|s = 1)$. Assuming $P(s = 1|x, a) \neq 0$ for all $t \in X \times A \times Y$, by the Bayes formula, we have

$$P_{\mathcal{Q}}(t) = \frac{P(t|s = 1)P(s = 1)}{P(s = 1|x, a)} = \frac{P(s = 1)}{P(s = 1|x, a)} P_{\mathcal{Q}_s}(t) \tag{5.1}$$

Hence, if we define and construct the new distribution $\hat{\mathcal{Q}}_s$ as $\frac{P(s=1)}{P(s=1|t)}\mathcal{Q}_s$, i.e., $P_{\hat{\mathcal{Q}}_s}(x,a,y) = \frac{P(s=1)}{P(s=1|x,a,y)}P_{\mathcal{Q}_s}(x,a,y)$, we have the following result.

$$
\begin{aligned}
&\mathbb{E}_{(x,a,y)\in\hat{\mathcal{Q}}_s}[l(f(x,a),y)] = \sum_{x,a,y} l(f(x,a),y))P_{\hat{\mathcal{Q}}_s}(x,a,y) \\
&= \sum_{x,a,y} l(f(x,a),y))\frac{P(s=1)}{P(s=1|x,a,y)}P_{\mathcal{Q}_s}(x,a,y) \\
&= \sum_{x,a,y} l(f(x,a),y))\frac{P(s=1)}{P(s=1|x,a,y)}P_{\mathcal{Q}}(x,a,y|s=1) \\
&= \sum_{x,a,y} l(f(x,a),y))\frac{P(s=1)}{P(s=1|x,a,y)}\frac{P_{\mathcal{Q}(s=1|x,a,y)}P_{\mathcal{Q}}(x,a,y)}{P_{\mathcal{Q}}(s=1)} \\
&= \sum_{x,a,y} l(f(x,a),y))P_{\mathcal{Q}}(x,a,y) \\
&= \mathbb{E}_{(x,a,y)\in\mathcal{Q}}[l(f(x,a),y)]
\end{aligned}
$$

Similarly we have $RD(\hat{\mathcal{Q}}_s) = RD(\mathcal{Q})$. Equivalently, if we define and construct a modified training dataset $\hat{\mathcal{D}}_s$ by introducing a weight $\frac{P(s=1)}{P(s=1|x,a,y)}$ to each record $t \in \mathcal{D}_s$, we can approximate $\mathbb{E}_{(x,a,y)\in\hat{\mathcal{Q}}_s}[l(f(x,a),y)]$ using $\mathbb{E}_{(x,a,y)\in\hat{\mathcal{D}}_s}[l(f(x,a),y)]$, which can be expressed as $\frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}}\frac{P(s=1)}{P(s=1|t)}l(f(x_i,a_i),y_i)$.

If we can derive the probabilities $P(s=1)$ and $P(s=1|x,a)$, we could derive the true probability $P_{\mathcal{Q}}$ from the biased one $P_{\mathcal{Q}_s}$ exactly and correct the sample selection bias in fair classification.

Theorem 1. Under sample selection bias, the classifier f that minimizes $\mathbb{E}_{(x,a,y,s)\in\hat{\mathcal{D}}_s}[l(f(x,a),y)]$ subject to $RD(\hat{\mathcal{D}}_s) \leq \tau$ is a fair classifier.

Note that the classifier f which tries to minimize $\frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}} l(f(x_i,a_i),y_i)$ under $RD(\mathcal{D}_s) \leq \tau$ would incur a large generalization error and cannot achieve fairness with respect to the true distribution $\mathcal{Q}$.

The sample selection bias causes training data to be selected non-uniformly from the population to be modeled. Generally there are four types of sample selection bias, missing completely at random when $P(s=1|x,a,y) = p(s=1)$, missing at random when $P(s=1|x,a,y) = p(s=1|x,a)$, missing at random-class when $P(s=1|x,a,y) = p(s=1|y)$, and missing not at random when there is no independence assumption between x, a, y and s. In the following, we focus on missing at random, which indicates biasedness of the selected sample depends on the feature vector x and protected attribute a.

The fairness constraint $RD(\hat{\mathcal{D}}_s) \leq \tau$ can be derived as

$$\left|\frac{\sum \mathbb{1}_{(x_i,a_i)\in\mathcal{D}_s^{11}} \frac{P(s=1)}{P(s=1|x_i,a_i)}}{\sum \mathbb{1}_{(x_i,a_i)\in\mathcal{D}_s^{\cdot 1}} \frac{P(s=1)}{P(s=1|x_i,a_i)}} - \frac{\sum \mathbb{1}_{(x_i,a_i)\in\mathcal{D}_s^{10}} \frac{P(s=1)}{P(s=1|x_i,a_i)}}{\sum \mathbb{1}_{(x_i,a_i)\in\mathcal{D}_s^{\cdot 0}} \frac{P(s=1)}{P(s=1|x_i,a_i)}}\right| \leq \tau, \tag{5.2}$$

where $\mathbb{1}_{[.]}$ is an indicator function, $\mathcal{D}_s^{ij} = \{(x_i,a_i)|\hat{Y}=i, A=j\}$ where $i,j \in \{0,1\}$, $\mathcal{D}_s^{\cdot j} = \{(x_i,a_i)|A=j\}$ where $j \in \{0,1\}$ and $\cdot$ represents $\{0,1\}$.

Then the minimization of the loss on $\hat{\mathcal{D}}_s$ subject to the fairness constraint is as:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w}) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} \frac{P(s=1)}{P(s=1|x_i,a_i)} l(f(x_i,a_i),y_i) \\ &\text{subject to} \qquad RD(\hat{D}_s) \leq \tau \end{aligned} \tag{5.3}$$

where $\mathbf{w}$ are the parameters of the classifier f and $R(\hat{D}_s)$ is shown in Equation 5.2. We then have our following corollary.

Corollary 1. With the assumption that selection variable s and label y are independent given x, i.e., $P(s|x,a,y) = P(s|x,a)$), the classifier f that minimizes $\mathbb{E}_{(x,a,y,s)\in\hat{\mathcal{D}}_s}[l(f(x,a),y)]$ subject to $RD(\hat{\mathcal{D}}_s) \leq \tau$ is a fair classifier under sample selection bias where $\hat{\mathcal{D}}_s$ is constructed by weighting each sample $t \in \mathcal{D}_s$ with $\frac{P(s=1)}{P(s=1|x,a)}$.

5.4 Robust Fairness-aware Learning

To obtain the optimal solution of Equation 5.3, we need to derive the sample selection probability $P(s=1|t)$. However, it is rather challenging to get the true $P(s=1|t)$ practically because the selection mechanism is usually unknown. Instead, we need to estimate the sample selection probability and use the estimated probability $\hat{P}(s=1|t)$ as the true $P(s=1|t)$.

To take the estimation error between $P(s=1|t)$ and $\hat{P}(s=1|t)$ into consideration, we adopt the approach of minimax robust minimization [48, 46, 47] which advocates for the worst case of any unknown true sample selection probability. We make an assumption here that the true $P(s=1|x,a)$ is with the ϵ range of the estimated $\hat{P}(s=1|x,a)$. Therefore, any value of $P(s=1|x,a)$ in this ϵ range represents the possible real unknown distribution $\mathcal{Q}$. Following the standard robust optimization approaches and taking the estimation error into consideration, we reformulate Equation 5.3:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathcal{W}} \max_{P(s=1|x_i,a_i)} L(\mathbf{w},P) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} \frac{P(s=1)}{P(s=1|x_i,a_i)} l(f(x_i,a_i),y_i) \\ &\text{subject to } |P(s=1|x_i,a_i) - \hat{P}(s=1|x_i,a_i)| \leq \epsilon \\ &\qquad RD(\hat{D}_s) \leq \tau \end{aligned} \tag{5.4}$$

In fact, $P(s=1)$ is a constant and does not affect the problem formulation and optimization. The robust minimax optimization can be treated as an adversarial game by two players. One player selects $P(s=1|x_i,a_i)$ within the ϵ range of the estimated $\hat{P}(s=1|x_i,a_i)$ to maximize the loss of the objective, which can be seen as the worst case of $\mathcal{Q}$. As aforementioned, the selection probability is determined by the corresponding distribution $\mathcal{Q}$ and then different selection probability can represent different $\mathcal{Q}$. Another player minimizes

the worst case loss to find the optimal $\mathbf{w}$. There are two advantages of robust minimax optimization of Equation 5.4. First, it takes the worst case induced by the estimation error into consideration, thus the obtained classifier f is robust to any possible $\mathcal{Q}$ within the error range of the estimation. Second, during the minimax optimization, the fairness is achieved under the worst case. Therefore, we can guarantee the fairness for any possible $\mathcal{Q}$ within the error range of the estimation.

The computation of $RD(\hat{D}_s)$ involves the indicator function, as shown in Equation 5.2, which makes it computationally intractable to reach the optimal solution of Equation 5.4. To address the intractable issue, we approximate the fairness constraint using the boundary fairness from Equation 3.3 in Chapter 3 and write the boundary fairness $C(t, \mathbf{w})$ on $\mathcal{D}_s$ as:

$$C_{\mathcal{D}_s}(t, \mathbf{w}) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} (a_i - \bar{a}) d_{\mathbf{w}(\mathbf{x}_i)}, \tag{5.5}$$

where a_i is the sensitive attribute value of t_i, $\bar{a} = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} a_i$ is the mean value of the sensitive attribute and $d_{\mathbf{w}(\mathbf{x}_i)}$ is the distance to the decision boundary of the classifier f and is formally defined as $d_{\mathbf{w}(\mathbf{x}_i)} = \mathbf{w}^T \mathbf{x}_i$. Similarly we will have the boundary fairness on $\hat{\mathcal{D}}_s$ as:

$$C_{\hat{\mathcal{D}}_s}(t, \mathbf{w}) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} (a_i - \bar{a}) \frac{P(s=1)}{P(s=1|x_i, a_i)} d_{\mathbf{w}(\mathbf{x}_i)}. \tag{5.6}$$

We enforce $C_{\hat{\mathcal{D}}_s}(t, \mathbf{w}) \leq \sigma, \sigma \in R^+$ to achieve the fair classification. With the boundary

fairness, we can rewrite the robust and fairness-aware loss function as:

$$\begin{aligned}
&\min_{\mathbf{w}\in\mathcal{W}} \max_{P(s=1|x_i,a_i)} L(\mathbf{w}, P) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} \frac{P(s=1)}{P(s=1|x_i,a_i)} l(f(x_i,a_i),y_i) \\
&\text{subject to } |P(s=1|x_i,a_i) - \hat{P}(s=1|x_i,a_i)| \leq \epsilon \\
&\qquad |\frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} (a_i - \bar{a}) \frac{P(s=1)}{P(s=1|x_i,a_i)} d_{\mathbf{w}(\mathbf{x}_i)}| \leq \sigma
\end{aligned} \tag{5.7}$$

5.4.1 Solving Robust Fairness-aware Optimization

The optimization of Equation 5.7 involves two sets of parameters $\mathbf{w}$ and $P(s = 1|x_i, a_i)$. According to its minimax formulation, it is preferable to obtain the optimal solution in an iterative manner by optimizing $\mathbf{w}$ and $P(s = 1|x_i, a_i)$ alternatively. First, we fix $P(s = 1|x_1, a_i)$ and decompose the part of Equation 5.7 only related to $\mathbf{w}$ as:

$$\begin{aligned}
&\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w}) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} \frac{P(s=1)}{P(s=1|x_i,a_i)} l(f(x_i,a_i),y_i) \\
&\text{subject to} \quad |\frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} (a_i - \bar{a}) \frac{P(s=1)}{P(s=1|x_i,a_i)} d_{\mathbf{w}(\mathbf{x}_i)}| \leq \sigma
\end{aligned} \tag{5.8}$$

It can be seen that with the fixed $P(s = 1|x_i, a_i)$, the optimization of $\mathbf{w}$ subjects to the linear fairness constraint. In fact, the optimal $\mathbf{w}$ is determined by the choice of loss function l and learning model f. For example, if the chosen model is regression model, then the optimization of $\mathbf{w}$ subject to the linear constraints belongs to the family of quadratic programming, which can be solved efficiently by the available tools. Instead, if we choose some complex models, e.g. deep learning models, then the loss function parameterized by $\mathbf{w}$ is non-convex, and it is quite challenging to apply the commonly used optimization techniques, e.g. gradient decent, to optimize $\mathbf{w}$ with the existence of linear constraints. Therefore, for the generalization

purpose, we choose to transform the fairness constraint as a penalty term and add to $L(\mathbf{w})$, which can be expressed as:

$$\begin{aligned}\min_{\mathbf{w}\in\mathcal{W}} L(\mathbf{w}) = \frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}}\frac{P(s=1)}{P(s=1|x_i,a_i)}l(f(x_i,a_i),y_i) \\ +\beta(\frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}}(a_i-\bar{a})\frac{P(s=1)}{P(s=1|x_i,a_i)}d_{\mathbf{w}(\mathbf{x}_i)}-\sigma)^2\end{aligned} \tag{5.9}$$

where β is a hyperparameter that controls the trade-off between the utility and fairness. By the transformation, standard optimization techniques such as stochastic gradient decent can be used to solve Equation 5.9.

Second, we can fix $\mathbf{w}$ and decompose the part of Equation 5.7 only related to $P(s=1|x_i,a_i)$ as the following:

$$\begin{aligned}\max_{P(s=1|x_i,a_i)} L(P) = \frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}}\frac{P(s=1)}{P(s=1|x_i,a_i)}l(f(x_i,a_i),y_i) \\ \text{subject to } |P(s=1|x_i,a_i)-\hat{P}(s=1|x_i,a_i)|\leq\epsilon \\ |\frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}}(a_i-\bar{a})\frac{P(s=1)}{P(s=1|x_i,a_i)}d_{\mathbf{w}(\mathbf{x}_i)}|\leq\sigma\end{aligned} \tag{5.10}$$

The objective of Equation 5.10 is a linear combination of $\frac{P(s=1)}{P(s=1|x_i,a_i)}$ as we can treat $\frac{P(s=1)}{P(s=1|x_i,a_i)}$ to be one variable. $P(s=1)$ is a constant and does not affect the optimization. For the first constraint, $|P(s=1|x_i,a_i)-\hat{P}(s=1|x_i,a_i)|\leq\epsilon$, we can obtain the range of $\frac{P(s=1)}{P(s=1|x_i,a_i)}$ after we estimate the range of each $P(s=1|x_i,a_i)$. The second constraint $|\frac{1}{N_{\mathcal{D}_s}}\sum_{i=1}^{N_{\mathcal{D}_s}}(a_i-\bar{a})\frac{P(s=1)}{P(s=1|x_i,a_i)}d_{\mathbf{w}(\mathbf{x}_i)}|\leq\sigma$ in Equation 5.10 is linear with respect to $\frac{P(s=1)}{P(s=1|x_i,a_i)}$ when $\mathbf{w}$ is fixed. Therefore, the optimization of $\frac{P(s=1)}{P(s=1|x_i,a_i)}$ is a standard linear programming and we can directly apply linear programming tool to get the optimal solution of $\frac{P(s=1)}{P(s=1|x_i,a_i)}$

without any additional relaxation.

5.4.2 *RFLearn*[1]: Sample Bias Correction with Test Data Available

In the previous section we formulate the robust and fairness-aware problem under the assumption that the estimated $\hat{P}(s = 1|x, a)$ is obtained, but without addressing how to estimate it. In this section, we assume the unlabeled $\mathcal{D}$ is available and estimate the true $P(s = 1|x, a)$ by using $\mathcal{D}$ and $\mathcal{D}_s$ together. For a particular data record t, the ratio between the number of times t in $\mathcal{D}$ and the number of times t in $\mathcal{D}_s$ in terms of (a, x) is an estimation value for $P(s = 1|x, a)$. Formally, for $t \in \mathcal{D}$, let $\mathcal{D}^t$ denote the subset of $\mathcal{D}$ containing exactly all the instances of t and $n_t = |\mathcal{D}^t|$. Similarly, let $\mathcal{D}_s^t$ denote the subset of $\mathcal{D}_s$ containing exactly all the instances of t and $m_t = |\mathcal{D}_s^t|$. We then have $\hat{P}(s = 1|x, a) = \frac{m_t}{n_t}$.

Lemma 1 [9] Let $\delta > 0$, then, with probability at least $1 - \delta$, the following inequality holds for all $t \in \mathcal{D}_s$:

$$|P(s = 1|x, a) - \frac{m_t}{n_t}| \leq \sqrt{\frac{\ln 2m' + \ln\frac{1}{\delta}}{p_0 N_{\mathcal{D}}}} \tag{5.11}$$

where m' denotes the number of unique points in $\mathcal{D}_s$ and $p_0 = \min_{t \in \mathcal{D}} P(t) \neq 0$.

For notation convenience, we define $\theta(x_i, a_i) = P(s = 1|x_i, a_i)$ and $\hat{\theta}(x_i, a_i) = \hat{P}(s = 1|x_i, a_i)$, where $\hat{\theta}(x_i, a_i)$ is the empirical value based on the frequency estimation. Lemma 1 states that $|\theta(x_i, a_i) - \hat{\theta}(x_i, a_i)|$ is upper bounded by the right term in Equation 5.11. Then we can apply the robust fairness aware framework by setting $\theta(x_i, a_i)$ within ϵ range of estimated $\hat{\theta}(x_i, a_i)$, where ϵ is the right term of Equation 5.11. According to the Theorem 2 in [9], the generalization error between the true distribution $\theta(x_i, a_i)$ and distribution using

Table 5.1: Model performance under data distribution shift (Adult and Dutch) Acc: accuracy

Methods	Adult Dataset				Dutch Dataset			
	Training Acc	Test Acc	Training *RD*	Test *RD*	Training Acc	Test Acc	Training *RD*	Test *RD*
LR (unbiased)	0.8124	0.8126	0.1562	0.1373	0.7493	0.7669	0.1498	0.1395
LR	0.8041	0.7882	0.1344	0.1228	0.7018	0.6821	0.0378	0.1012
FairLR	0.7823	0.7622	0.0231	0.0956	0.7006	0.6624	0.0289	0.0991
[47]	0.7883	0.8048	0.1348	0.1333	0.6812	0.7044	0.1421	0.1394
$RFLearn^{1-}$	0.7412	0.7875	0.0351	0.1048	0.6501	0.6879	0.0315	0.0809
$RFLearn^{1}$	0.7484	0.7816	0.0281	0.0416	0.6673	0.6910	0.0317	0.0405
$RFLearn^{2-}$	0.7473	0.7771	0.0321	0.0963	0.6457	0.6809	0.0411	0.0973
$RFLearn^{2}$	0.7336	0.7678	0.0197	0.0238	0.6479	0.6755	0.0321	0.0373

Table 5.2: Model performance of $RFLearn^{1}$ under sample selection bias with different δ (Adult and Dutch). Acc: accuracy

δ	Adult Dataset				Dutch Dataset			
	Training Acc	Test Acc	Training *RD*	Test *RD*	Training Acc	Test Acc	Training *RD*	Test *RD*
0.025	0.7181	0.7601	0.0189	0.0219	0.6521	0.6812	0.0291	0.0326
0.05	0.7217	0.7673	0.0239	0.0398	0.6521	0.6812	0.0321	0.0326
0.1	0.7484	0.7816	0.0307	0.0416	0.6673	0.6910	0.0378	0.0405
0.15	0.7239	0.7768	0.0277	0.0333	0.6701	0.6994	0.0275	0.0379

the estimated $\hat{\theta}(x_i, a_i)$ is expressed as:

$$|L_\theta(\mathbf{w}) - L_{\hat{\theta}}(\mathbf{w})| < \mu\sqrt{\frac{\ln 2m' + \ln\frac{1}{\delta}}{p_0 N_{\mathcal{D}}}} \tag{5.12}$$

where μ is a constant determined by σ (Lemma 1) and hyperparameter β (Equation 5.9). Suppose the maximum value of $L_{\hat{\theta}}(\mathbf{w})$ is defined as $L_{\hat{\theta}}(\mathbf{w})_{max}$ and our robust fairness-aware optimization is to minimize $L_{\hat{\theta}}(\mathbf{w})_{max}$ per iteration. The loss $L_{\hat{\theta}}(\mathbf{w})_{max}$ consists of both the prediction loss and fairness loss. Therefore, the minimization of upper bound of the generalization error of true distribution can provide robustness in terms of both prediction and fairness.

Table 5.3: Model performance of *RFLearn*^{2} under sample selection bias with different ρ (Adult and Dutch). Acc: accuracy

ρ	**Adult Dataset**				**Dutch Dataset**			
	Training Acc	Test Acc	Training *RD*	Test *RD*	Training Acc	Test Acc	Training *RD*	Test *RD*
0.2	0.7229	0.7558	0.0178	0.0114	0.6401	0.6543	0.0175	0.0214
0.4	0.7336	0.7628	0.0197	0.0238	0.6479	0.6755	0.0321	0.0373
0.6	0.7428	0.7724	0.0269	0.0361	0.6544	0.6792	0.0301	0.0314

5.4.3 *RFLearn*^{2}: Sample Bias Correction without Test Data Available

In this section, we focus on the scenario without unlabeled test data $\mathcal{D}$. The challenge is how to use $\mathcal{D}_s$ alone to estimate the true sample selection probability so that we can construct $\hat{\mathcal{D}}_s$ to resemble $\mathcal{D}$. In general, the exact relationship between $\mathcal{D}_s$ and $\mathcal{D}$ is unknown. Without additional assumptions, it is impossible to build a model to resemble the true $\mathcal{D}$ with only access to the observed $\mathcal{D}_s$.

We assume that 1) there exist K clusters in $\mathcal{D}_s$; 2) the samples in the same cluster have the same selection probability; and 3) the selection probability of each sample is within a range of the uniform selection probability. Under these assumptions, the ratio $\frac{P(s=1)}{P(s=1|x_i,a_i)}$ for each sample from the same cluster is the same. The ratio vector for K clusters is defined as $\mathbf{r} = (r_1, r_2, \cdots, r_K)$. We robustify the estimation by approximating r within a Wasserstein ball B_ρ [92] around the uniform ratio $\mathbf{r}_u$, where all of the values in $\mathbf{r}_u$ is 1. Formally we have $|\mathbf{r} - \mathbf{r}_u| \leq \rho$, where ρ is the radius of the Wasserstein ball. Suppose x_i belongs to the k-th cluster where $k \in [K]$, we can write the robust loss of f with only $\mathcal{D}_s$ available as the

following:

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{r}\in B_\rho} L(\mathbf{w},\mathbf{r}) = \frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} r_k l(f(x_i,a_i),y_i)$$

$$\text{subject to} \qquad |\mathbf{r}-\mathbf{r}_u| \le \rho \tag{5.13}$$

$$|\frac{1}{N_{\mathcal{D}_s}} \sum_{i=1}^{N_{\mathcal{D}_s}} (a_i-\bar{a}) r_k d_{\mathbf{w}(\mathbf{x}_i)}| \le \sigma$$

The above formulation also saves huge amounts of computational cost as it reduces the number of constraints from the training data size level to the cluster size level.

5.5 Experiments

Dataset. We use two benchmark datasets, Adult [90] and Dutch [28], to evaluate our proposed algorithms, *RFLearn*^{1} and *RFLearn*^{2}. Adult dataset consists of individual's information such as age, education level, gender, occupation, race, and so forth. The task for Adult dataset is to predict whether an individual's income is over 50k or not based on the collected personal information. Dutch dataset is a collection of Census data in Netherlands, where each data sample has different attributes like Adult datasets, including education level, gender, race, marital status and so forth. The task is also a binary classification to predict whether an individual belongs to low income or high income group. In both datasets, we set *gender* as the protected attribute and use one-hot encoding to convert the categorical attributes to vectors and apply normalization to covert numerical attributes into the range $[0,1]$. After preprocessing, Adult has 45222 records and 40 features, while Dutch has 60420 records and 35 features.

Experimental Setting. We follow the biased data generation approach in [93] and choose to select the data based on the education level (married status) for Adult (Dutch). For Adult, we create the biased training data by selecting 18157 records from the first 35000 records

(randomly choosing 70% of people with 10+ years education and 30% of people with less than 10 years education) and use the rest 10222 records as the test data. Similarly for Dutch, we create the biased training data by selecting 20928 examples from the first 45000 records (randomly choosing 75% of married people and 25% of unmarried people) and use the rest 15420 records as the test data.

Baselines. We choose the logistic regression (LR) to implement and evaluate different algorithms. We consider the following baselines to compare with our proposed $RFLearn^1$ and $RFLearn^2$: (a) LR without fairness constraint (LR); (b) LR with fairness constraint ($FairLR$); (c) robust LR in [47] that uses kernel functions to reweigh samples under covariate shift but ignores the fairness constraint. For $RFLearn^1$ ($RFLearn^2$), we also consider its variation $RFLearn^{1\text{-}}$ ($RFLearn^{2\text{-}}$) that optimizes the robust loss with unweighted fairness constraint.

Metrics and Hyperparameters. We evaluate the performance in terms of prediction accuracy and risk difference (RD). We consider a classifier is fair if $RD(f) \leq 0.05$. The hyperparameter β that controls the accuracy-fairness trade-off is set as 1. The radius ρ is set as 0.4, δ is set as 0.1, and σ is set as 0.2. For $RFLearn^{2-}$ and $RFLearn^2$, we apply K-means [94] to cluster the training data and set the number of clusters 300. We run all experiments 20 times and report the average results.

5.5.1 Accuracy vs. Fairness

We report the accuracy and fairness on both training data and test data for each method in our results. Note that our goal is to achieve robust accuracy and fairness on test data under the sample selection bias. Results on both training and test data help explain why baselines focusing on the training data itself fail achieving good accuracy and fairness on test data.

Accuracy. Table 5.1 shows prediction accuracy and risk difference of each model on training and test data for both Adult and Dutch. The first row *LR (unbiased)* indicates the performance of the logistic regression model with unbiased training data. We randomly select the same number of data from the original datasets. The results of the rest rows are all conducted on our biased training data discussed in the experimental setting section. We can draw several important points from the experimental results.

First, the testing accuracy of *LR* is 0.7882 for Adult and 0.6821 for Dutch, whereas the accuracy of *LR (unbiased)* is 0.8126 for Adult and 0.7669 for Dutch. It demonstrates that the model prediction performance degrades under sample selection bias. Second, with the use of robust learning, the prediction accuracy for Adult (Ducth) of $RFLearn^{1-}$ is 0.7875 (0.6879) and $RFLearn^{2-}$ is 0.7771 (0.6809), which outperforms *FairLR* by 0.0253 (0.0255) and 0.0109 (0.0185). As the fairness constraints of $RFLearn^{1-}$ and $RFLearn^{2-}$ are the same with *FairLR*, it demonstrates that the robust learning in Equation 5.7 can provide robust prediction under the sample selection bias. Third, the testing accuracy from robust learning methods is higher than the training accuracy, which further demonstrates the advantage of robust learning under the sample selection bias. Fourth, the accuracy of $RFLearn^{1}$ is higher than that of $RFLearn^{2}$. It is reasonable as we leverage the unlabeled test data in our $RFLearn^{1}$.

Fairness. In this section, we focus on fairness comparison. We summarize below the interesting observations from Table 5.1. First, all of *FairLR*, $RFLearn^{1-}$ and $RFLearn^{2-}$ can only achieve fairness on the training data with $RD \leq 0.05$, but none of these approaches can guarantee the fairness on the test data. $RFLearn^{1-}$ and $RFLearn^{2-}$, which uses the unweighted fairness constraint, also fail to achieve fairness on the test data. The method proposed by [47] only considers the robustness of prediction error but ignores the fairness, so that it cannot

achieve the fairness on the test data as well. Second, $RFLearn^1$ and $RFLearn^2$ can achieve fairness on both training and test data. The result is consistent with our algorithms as we enforce the fairness under any possible adverse distribution. Therefore, the classifier is also robust in terms of meeting fairness requirement on the test data even it is trained on the biased sampling data.

5.5.2 Effects of Hyperparameters

In this section, we conduct sensitivity analysis of our $RFLearn^1$ and $RFLearn^2$ with different hyperparameters. Due to space limits, we focus on two key parameters δ and ρ. Table 5.2 shows the performance of $RFLearn^1$ with different δ values. Note that in Lemma 1 the estimation error of the sample selection probability is upper bounded with the probability greater than $1 - \delta$. The upper bound (the right side in Equation 5.11) increases with the decreasing δ. A larger upper bound indicates that the adversary can generate more possible distributions during the robust optimization, hence helping achieve better prediction accuracy on test data. However, when the upper bound is too large, excessive possible distributions may reduce the prediction accuracy on test data. The experimental results in Table 5.2 match our above analysis.

Table 5.3 shows the result of $RFLearn^2$ under different radius ρ. We can see that the testing accuracy increases with the increasing ρ. Larger ρ indicates more possible generated distributions which are more likely to cover the test distribution and improve the model performance. Moreover, the proposed $RFLearn^1$ and $RFLearn^2$ can achieve both fairness on the training and test data with different δ and ρ. It is beneficial as the fairness performance of our proposed algorithms are not sensitive to the varying δ and ρ.

5.6 Summary

In this chapter we have developed a robust and fair learning framework with two algorithms to deal with the sample selection bias. Our framework adopts the reweighing estimation approach for bias correction and the minimax robust estimation for achieving robustness on prediction accuracy and fairness on test data.

6 Achieve Regression Fairness under Sample Selection Bias

6.1 Introduction

Fairness has been an increasingly important topic in machine learning. Fair machine learning models aim to learn a function f for a target variable Y using input features X and a sensitive attribute A (e.g., gender), while ensuring the predicted value $\hat{Y}$ fair with respect to A based on some given fairness criterion. Fair machine learning models can be categorized into pre-processing (modifying training data or learning a new representation such that the information correlated to the sensitive attribute is removed), in-processing (adding fairness penalty to the objective function during training), and post-processing (applying perturbation or transformation to model output to reduce prediction unfairness). Much of existing works has focused on classification. In this paper, we focus on fair regression where the target Y is continuous.

Fair regression can be naturally defined as the task of minimizing the expected loss of real-valued predictions, subject to some fairness constraints. Fairness notions under the regression setting are in principle based on some forms of independence, e.g., the independence of model prediction $\hat{Y}$ and sensitive attribute A, the independence of prediction error $\hat{Y} - Y$ and sensitive attribute A, and the conditional independence of $\hat{Y}$ and A given Y. Different from the classification setting, variables of Y and $\hat{Y}$ (even A) become continuous in the regression setting, which requires new fairness notions and constrained optimization techniques. Researchers have developed quantitative metrics based on moment constraints, such as mean difference [95], mean squared error difference [96], and Pearson correlation [97]. These simplified metrics can be easily calculated but fail to capture subtle effects. For exam-

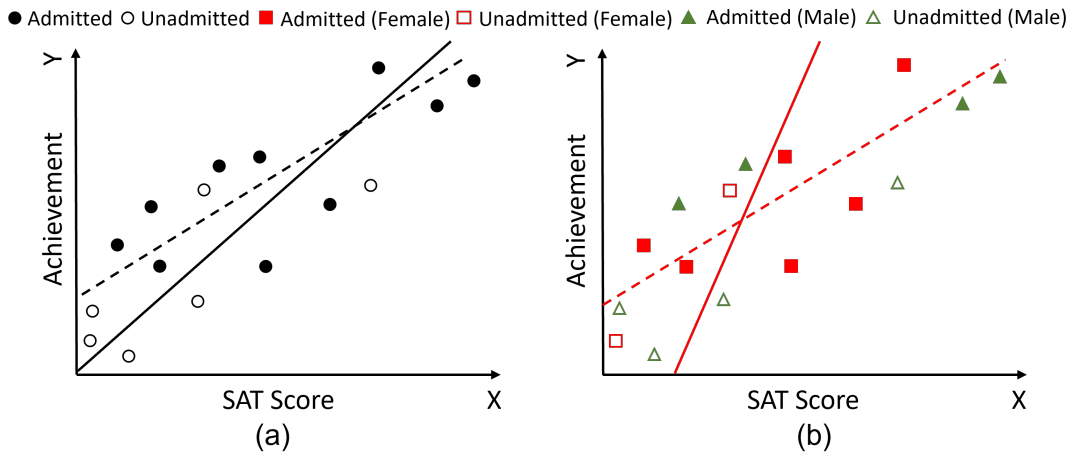

Figure 6.1: Illustrative example for fair regression under sample selection bias. Fitted models with sample correction incorporate feature values of unadmitted students. Black line: LR w/o correction, black dashed line: LR with correction, red line: fair LR w/o correction, red dashed line: fair LR with correction.

ple, the predicted values may have different variances across groups. Recently, researchers started to propose fairness metrics based on distributions/densities instead of simple point estimate [98], and develop approximation methods [99] for achieving fairness in regression. It is imperative to develop a general fair regression framework that enforces a variety of fairness notions and provides efficient implementation and theoretical analysis when dual optimization and approximation are applied. Moreover, all previous fair regression research assumed the training data and testing data are drawn from the same distributions. This assumption is often violated in real world due to the sample selection bias between the training and testing data.

Figure 6.1 shows an illustrative example of studying the relationship between SAT scores (X) and potential college achievement (Y) of students. The regression model trained on only observed student samples who were already admitted to college (denoted as $\mathcal{D}_s$ and shown as solid data points in Figure 6.1(a)) would be biased as the fitted model did not consider applicants who could potentially go to college (denoted as $\mathcal{D}_u$ and shown as hollow

points in Figure 6.1(a)). Note that for these applicants who did not go to college, SAT scores (X) are still available although their corresponding college achievements (Y) are missing. Moreover, as shown in Figure 6.1(b), the fair regression model trained on only admitted college students D_s in fact would be unfair and cannot be adopted for future applicants whose distribution is assumed to resemble the union of D_s and D_u. It is imperative to learn a fair regression model that can incorporate X values of samples from $\mathcal{D}_u$ to both improve model fitness and achieve fairness on population.

In this chapter, we propose, *FairLR**, the fair regression framework under sample selection bias when dependent variable values of a set of samples from the training data are missing as a result of another hidden selection process. Our *FairLR** adopts the classic Heckman model [21] for bias correction and the Lagrange duality theory [22] to achieve regression fairness based on a variety of fairness notions. Our fair regression framework minimizes the loss function subject to fairness inequality and equality constraints. We apply the Lagrange duality theory to transform the primal problem into a dual convex optimization problem. For the two popular fairness notions, mean difference (MD) and mean squared error difference (MSED), we derive two explicit formulas without optimizing iteratively. For Pearson correlation, we derive its conditions of satisfying the Slater condition, thus achieving strong duality. We conduct experiments on three real-world datasets and the experimental results demonstrate our approach's effectiveness in terms of both utility and fairness. Note that this chapter is originally from the arxiv version in [100].

6.2 Related Work

Fair Regression. For linear regression $f(\cdot) : X \to Y$ with discrete sensitive attribute A, Calders et al. [95] first introduced mean difference and AUC to measure the unfairness.

Fitzsimons et al. [101] also used a similar concept termed as group fairness expectation to ensure fair prediction for different groups. For regression with discrete/continuous sensitive attribute, Mary et al. [102] used the Rényi maximum correlation coefficient of prediction and sensitive attribute to describe the fairness penalty. Recently, Agarwal et al. [98] presented two fairness definitions, statistical parity and bounded group loss. The statistical parity uses the departure of the CDF of $f(X)$ conditional on $A = a$ from the CDF of $f(X)$. When the departure is close to zero, the prediction is statistically independent of the protected attribute. The bounded group loss requires that the prediction error of any protected group stay below some pre-determined threshold.

To address the challenge of estimating information-theoretic divergences between conditional probability density functions, Steinberg et al. [99] introduced fast approximations of the independence, separation and sufficiency group fairness criteria for regression models from their (conditional) mutual information definitions. Chzhen et al. [103] focused on demographic parity that requires the distribution of the predicted output to be independent of the sensitive attribute. They established a connection between fair regression and optimal transport theory and derived a closed form expression for the optimal fair predictor, i.e., the distribution of this optimum is the Wasserstein barycenter of the distributions induced by the standard regression function on the sensitive groups.

Fair Classification under Sample Selection Bias. The sample selection bias causes training data to be selected non-uniformly from the population to be modeled. Extensive research has been conducted on classification under sample selection bias (refer to a survey [87]). Some recent research focused on robust classification under sample selection bias [48, 46, 47, 61, 104]. For example, Wen et al. [47] considered covariate shift between the training and testing data and proposed a minimax robust framework that applies Gaussian kernel

functions to reweigh the training examples. Du et al. [104] adopted the reweighing estimation idea for sample bias correction and used the minimax robust estimation to achieve robustness on prediction accuracy. To tackle the unfairness issue under distribution shift, Taskesen et al. [51] developed a distributionally robust logistic regression model with an unfairness penalty. They assumed the unknown true test distribution is contained in a Wasserstein ball centered at the empirical distribution on the observed training data. Rezaei et al. [52] proposed the use of ambiguity set to derive the fair classifier based on the principles of distributional robustness. Other related work includes fair transfer learning [54] and fair classification with bias in label collection [57, 59].

6.3 Fair Regression under Sample Selection Bias

6.3.1 Problem Formulation

We first define the notations used in this paper. Let $(\boldsymbol{X}, A, Y)$ denote the training data $\mathcal{D}$, where $\boldsymbol{X}$ is the feature space, A is the protected attribute, and Y is the continuous target attribute. $\mathcal{D}$ contains n data samples, among which m samples $(\boldsymbol{x}_i, a_i, y_i)$ are fully observed and the remaining $n-m$ data points have y_i missing. We denote the fully observed part as $\mathcal{D}_s$ and the other part as $\mathcal{D}_u$. The whole training data $\mathcal{D}$ is selected uniformly from the population to be modeled. However, $\mathcal{D}_s$ is non-uniformly selected and the bias of $\mathcal{D}_s$ could depend on both feature vector $\boldsymbol{x}, a$ and target variable y. A regression function $f(\cdot) : X \to Y$ tries to learn optimal parameter $\boldsymbol{w}$. We denote $\hat{y} = f(\boldsymbol{x}; \boldsymbol{w})$.

Problem Statement. Given the training dataset $\mathcal{D} = \mathcal{D}_s \cup \mathcal{D}_u$, derive regression function $f(\cdot) : X \to Y$ that achieves fairness on population with respect to protected attribute A based on some fairness criterion.

We emphasize the sample selection bias considered in our work is different from the traditional covariate shift scenario. Although the covariate shift tackles the shift between the training distribution $P_{tr}(X)$ and test distribution $P_{te}(X)$, it usually assumes both a labeled training dataset and an unlabeled testing dataset are available in the training phase. In our work, we do not require the unlabeled testing data in the training phase. Instead, we assume the available training dataset contains a mixture of labeled and unlabeled data points but the labeling process is biased. In our setting, we are able to use the Heckman model to correct the bias with theoretical guarantee as we can compute the conditional unbiased expectation analytically. However, the previous commonly used approaches for covariate shift can only achieve robust estimation within a range but cannot provide theoretical guarantee.

6.3.2 Heckman Model Revisited

Heckman model [21] addresses the issue of sample selection bias when the dependent variable in the regression has values that are missing not at random. In the two-step estimation procedure of Heckman model, the first step uses probit regression to model the sample selection process and derives a new variable called the Inverse Mills Ratio (IMR). The second step adds the IMR to the regression analysis as an independent variable and uses ordinary least squares to estimate the regression coefficients. This two-step estimator can perform well when there is no multicollinearity between the IMR and the explanatory variables. We present below the Heckman model formally.

The selection equation of the ith sample is $z_i = \boldsymbol{x}_{1i}\boldsymbol{\gamma} + u_i$ where $\boldsymbol{x}_{1i}$ includes the set of features related to sample selection, $\boldsymbol{\gamma}$ is the set of regression coefficients, and u_i is the

error term. The selection index s is defined as:

$$s_i = \begin{cases} 1 & z_i > 0 \\ \\ 0 & z_i \leq 0 \end{cases} \tag{6.1}$$

where $s_i = 1$ indicates that the ith sample is fully observed and $s_i = 0$ indicates its target value y_i is missing. The prediction model is based on linear regression and for the ith sample we have $y_i = \hat{y}_i + \epsilon_i = \boldsymbol{x}_{2i}\boldsymbol{\beta} + \epsilon_i$ where $\hat{y}_i$ is the predicted value, $\boldsymbol{x}_{2i}$ includes the set of features used for prediction, $\boldsymbol{\beta}$ is the set of regression coefficients, and ϵ_i is the error term. Following the default assumptions in the Heckman model, $\boldsymbol{x}_{2i}$ is a subset of $\boldsymbol{x}_{1i}$, indicating that all attributes predicting the outcome of interest can also predict selection equation, and $u_i \sim N(0,1)$, and $\epsilon_i \sim N(0, \sigma_\epsilon^2)$. The correlation coefficient of u_i and ϵ_i is denoted by ρ. The prediction outcome based on $\mathcal{D}_s$ alone is biased and we can correct it by computing the conditional means of the prediction outcome as:

$$\begin{aligned} \mathbb{E}(y_i|s_i = 1) = \mathbb{E}(y_i|z_i > 0) &= \mathbb{E}(\boldsymbol{x}_{2i}\boldsymbol{\beta} + \epsilon_i|\boldsymbol{x}_{1i}\boldsymbol{\gamma} + u_i > 0) \\ &= \boldsymbol{x}_{2i}\boldsymbol{\beta} + \mathbb{E}(\epsilon_i|\boldsymbol{x}_{1i}\boldsymbol{\gamma} + u_i > 0) \\ &= \boldsymbol{x}_{2i}\boldsymbol{\beta} + \mathbb{E}(\epsilon_i|u_i > -\boldsymbol{x}_{1i}\boldsymbol{\gamma}) \end{aligned} \tag{6.2}$$

Because u_i and ϵ_i are correlated, then we have (see the supplemental material for derivations)

$$\mathbb{E}(\epsilon_i|u_i > -\boldsymbol{x}_{1i}\boldsymbol{\gamma}) = \alpha_i \rho \sigma_\epsilon \tag{6.3}$$

where $\alpha_i = \dfrac{\phi(-\boldsymbol{x}_{1i}\boldsymbol{\gamma})}{1 - \Phi(-\boldsymbol{x}_{1i}\boldsymbol{\gamma})}$ is usually termed as IMR. Here $\phi(\cdot)$ denotes the standard normal density function and $\Phi(\cdot)$ denotes the standard cumulative distribution function.

To compute the value of α_i, the first step is to estimate the coefficients $\boldsymbol{\gamma}$. We use the

maximum likelihood estimate (MLE) to estimate $\boldsymbol{\gamma}$ by treating the selection equation as a probit classification model and we have

$$P(s_i = 1) = \Phi(\boldsymbol{x}_{1i}\boldsymbol{\gamma}), P(s_i = 0) = 1 - \Phi(\boldsymbol{x}_{1i}\boldsymbol{\gamma}) \tag{6.4}$$

Then the likelihood of $\mathcal{D}$ is expressed as:

$$LH(\boldsymbol{\gamma}; s_i, \boldsymbol{x}_{1i}) = \prod_{i=1}^{n} \Phi(\boldsymbol{x}_{1i}\boldsymbol{\gamma})^{s_i}(1 - \Phi(\boldsymbol{x}_{1i}\boldsymbol{\gamma}))^{1-s_i} \tag{6.5}$$

The maximization of Equation 6.5 will obtain the estimates of $\boldsymbol{\gamma}$, and thus we can compute α_i for each selected sample (x_i, a_i, y_i) in D_s. With available α_i, we can rewrite Equation 6.2 as:

$$\mathbb{E}(y_i|s_i = 1) = \boldsymbol{x}_{2i}\boldsymbol{\beta} + \alpha_i\rho\sigma_\epsilon \tag{6.6}$$

Then we can estimate the coefficients $\boldsymbol{\beta}$ from Equation 6.6, e.g., via the ordinary least squares (OLS) by minimizing $\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \sum_{i=1}^{m}(\boldsymbol{x}_{2i}\boldsymbol{\beta} + \alpha_i\rho\sigma_\epsilon - y_i)^2$.

6.3.3 Fair Regression via Heckman Correction

We first present the general fair regression framework that aims to minimize the risk and learns the parameters $\boldsymbol{w} \in \mathcal{W}$ subject to the fairness constraints:

$$\begin{aligned} &\min_{\boldsymbol{w}\in\mathcal{W}} \mathbb{E}[l(\hat{y}, y)] = \mathbb{E}[l(f(\boldsymbol{x}; \boldsymbol{w}), y)] \\ &\text{subject to} \quad g_i(\hat{y}, y, a) \leq 0, i = 1, \cdots, p \\ &\qquad\qquad h_j(\hat{y}, y, a) = 0, j = 1, \cdots, q \end{aligned} \tag{6.7}$$

where f is the learning model, l is the loss function, g_i are fairness inequality constraints, and h_j are fairness equality constraints.

We then formulate the fair regression under sample selection bias and rewrite Equation 6.7 to minimize the empirical loss subject to the fairness constraint:

$$\begin{aligned} p^* = \min_{\tilde{\boldsymbol{\beta}}} L(\tilde{\boldsymbol{\beta}}) &= \sum_{i=1}^{m} l[(f_h(\tilde{\boldsymbol{x}}_{2i}; \tilde{\boldsymbol{\beta}}), y)] \\ \text{subject to} \quad g_i(\tilde{\boldsymbol{\beta}}) &\leq 0, i = 1, \cdots, p \\ h_j(\tilde{\boldsymbol{\beta}}) &= 0, j = 1, \cdots, q \end{aligned} \tag{6.8}$$

where $\tilde{\boldsymbol{\beta}} = [\boldsymbol{\beta}, \boldsymbol{\beta}_\alpha]$, $\beta_\alpha = \rho\sigma_\epsilon$, $\tilde{\boldsymbol{x}}_{2i} = [\boldsymbol{x}_{2i}, \alpha_i]$, and $f_h(\tilde{\boldsymbol{x}}_{2i}; \tilde{\boldsymbol{\beta}}) = \tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}}$ is the Heckman prediction function. Note that from Equation 6.6 the bias is corrected by α_i which carries the information of $\mathcal{D}_u$. The effect of each α_i on the sample (x_i, a_i, y_i) is quantified by $\rho\sigma_\epsilon$. We can then treat $\rho\sigma_\epsilon$ as one additional dimension β_α of the coefficient vector. In addition, the fairness constraint computed based on the corrected $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{x}}_{2i}$ is unbiased.

6.3.3.1 Fairness Notions

Previous works on fair regression developed notions are based on the independence of model prediction (or prediction error) and sensitive attribute. Different from the classification setting, target variable Y is continuous and sensitive attribute A can be either categorical or continuous. Table 6.1 summarizes fairness notions including their formula, reference, and applicability in terms of sensitive attribute type. We leave their formal definitions in the supplemental material. The mean difference (MD), the mean squared error difference (MSED), the statistical parity (SP), and the bounded group loss (BGL) handle categorical sensitive attribute whereas Pearson correlation ($\rho_{\hat{y}a}$) and our introduced partial correlation ($\rho_{\hat{y}a.y}$)

handles numerical sensitive attribute. The partial correlation $\rho_{\hat{y}a.y}$ includes both y and a in the condition which is similar to the equalized opportunity [14] in fair classification. MD, Pearson and SP focus on independence of model prediction and sensitive attribute whereas MSED, Partial and BGL consider prediction error. Moreover, SP (BGL) measures the dependence of prediction (prediction error) and sensitive attribute on distributions/densities, in contrary to point estimate of other notions.

Table 6.1: Fairness notions for regression

Definition	Reference	Equation	Categorical	Numeric
MD	[95, 96, 105, 106, 107, 108]	$\text{MD}(\hat{y}, a) = \mathbb{E}(\hat{y}\|a = 0) - \mathbb{E}(\hat{y}\|a = 1)$	✓	×
MSED	[95, 96]	$\text{MSED}(\hat{y}, a) = \mathbb{E}[(y - \hat{y})^2\|a = 0] - \mathbb{E}[(y - \hat{y})^2\|a = 1]$	✓	×
Pearson	[97]	$\rho_{\hat{y}a} = \frac{\mathbb{E}[(\hat{y}-\mu_{\hat{y}})(a-\mu_a)]}{\sigma_{\hat{y}}\sigma_s}$	×	✓
Partial	ours	$\rho_{\hat{y}a.y} = \frac{\rho_{\hat{y}a}-\rho_{\hat{y}y}\rho_{ay}}{\sqrt{1-\rho^2_{\hat{y}y}}\sqrt{1-\rho^2_{ay}}}$	×	✓
SP	[98, 105, 108, 109]	$\text{SP} = \mathbb{P}[f(X) \geq z\|A = a] - \mathbb{P}[f(X) \geq z]$	✓	×
BGL	[98, 110]	$\text{BGL} = \mathbb{E}[l(f(X), Y)\|A = a]$	✓	×

In general, we can enforce strict fairness via equality constraints and relaxed fairness via inequality constraints in Equation 6.8. For example, we use $h_j(\tilde{\boldsymbol{\beta}}) = 0$ for $\text{MD} = 0$, and use $g_i(\tilde{\boldsymbol{\beta}}) - \tau \leq 0$ for $\text{MD} \leq \tau$ where τ is a user-specified threshold. One challenge is for SP as the number of constraints is uncountable. We can apply the algorithm developed in [98] that discretizes the real-valued prediction space and reduces the optimization problem to cost-sensitive classification. The cost-sensitive classification is then solved by the reduction approach [111]. We note that our framework can be used to enforce multiple fairness notions at the same time and some notions may be mutually contradictory [112], which can cause vacuous solutions.

6.3.3.2 Dual Formulation

To solve the primal optimal problem (Equation 6.8) with a variety of fairness notions, we apply the Lagrange duality theory [22] to relax the primal problem by its constraints. The Lagrangian function is

$$L_c(\tilde{\beta}, \lambda, \upsilon) = L(\tilde{\beta}) + \lambda^T g(\tilde{\beta}) + \upsilon^T h(\tilde{\beta}) \tag{6.9}$$

where $\lambda \in \mathbb{R}_+^p$ and $\upsilon \in \mathbb{R}^q$ are the Lagrange multiplier vectors (or dual variables) associated with inequality constraints and equality constraints. The dual function hence is defined as $Q(\lambda, \upsilon) = \inf_{\tilde{\beta}} L_c(\tilde{\beta}, \lambda, \upsilon)$. Note that the dual function $Q(\lambda, \upsilon)$ is a pointwise affine function of (λ, υ), it is concave even when the problem (Equation 6.8) is non-convex. For each pair (λ, υ), the dual function gives us a lower bound of the optimal value p^*, i.e., $Q(\lambda, \upsilon) \leq p^*$. The best lower bound leads to the Lagrange dual problem:

$$d^* = \max_{\lambda \succeq 0, \upsilon} Q(\lambda, \upsilon) = \max_{\lambda \succeq 0, \upsilon} \min_{\tilde{\beta}} L_c(\tilde{\beta}, \lambda, \upsilon) \tag{6.10}$$

The Lagrange dual problem is a convex optimization problem because the objective to be maximized is concave and the constraint is convex. We can solve the dual optimization problem by alternating gradient descent steps over the primal variables $\tilde{\beta}$ and dual variables (λ, υ), respectively. In particular, by iteratively executing the following two steps: 1) find $\tilde{\beta}^* \leftarrow argmin_{\tilde{\beta}} L_c(\tilde{\beta}, \lambda, \upsilon)$; 2) compute $\lambda \leftarrow \lambda + \eta \frac{dL_c}{d\lambda}(\tilde{\beta}^*, \lambda, \upsilon)$, $\upsilon \leftarrow \upsilon + \eta \frac{dL_c}{d\upsilon}(\tilde{\beta}^*, \lambda, \upsilon)$, the solution will converge.

Next, we show instantiations of two widely used fairness notions, MD and MSED, by deriving their explicit formulas without iterative optimization.

Result 1. For fair regression with the mean squared loss and $\mathrm{MD}(\hat{y}, a) = 0$, we have the closed solution

$$\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{X}}_2^T \tilde{\boldsymbol{X}}_2)^{-1}(\tilde{\boldsymbol{X}}_2^T \boldsymbol{y} - \frac{\boldsymbol{d}^T(\tilde{\boldsymbol{X}}_2^T \tilde{\boldsymbol{X}}_2)^{-1}\tilde{\boldsymbol{X}}_2^T \boldsymbol{y}}{\boldsymbol{d}^T(\tilde{\boldsymbol{X}}_2^T \tilde{\boldsymbol{X}}_2)^{-1}\boldsymbol{d}}\boldsymbol{d}) \tag{6.11}$$

Proof. The fair Heckman prediction model can be described as:

$$\begin{aligned} &\min L(\tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{m}(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2 \\ &\text{subject to} \quad \frac{1}{m_0}\sum_{i\in\mathcal{D}_0}\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} = \frac{1}{m_1}\sum_{i\in\mathcal{D}_1}\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} \end{aligned} \tag{6.12}$$

where m_0 is the number of data in $\mathcal{D}_s$ with $a = 0$, m_1 is the number of data with $a = 1$, and $m = m_0 + m_1$.

We solve this optimization problem Equation 6.12 using Lagrange multipliers. For convenience, we use $\boldsymbol{d}$ to denote $\frac{1}{m_0}\sum_{i\in\mathcal{D}_0}\tilde{\boldsymbol{x}}_{2i} - \frac{1}{m_1}\sum_{i\in\mathcal{D}_1}\tilde{\boldsymbol{x}}_{2i}$. Then we can rewrite Equation 6.12 as the following constrained minimization problem:

$$L(\tilde{\beta}) = \min\sum_{i=1}^{m}(\tilde{\boldsymbol{\beta}}\tilde{\boldsymbol{x}}_{2i} - y_i)^2 + 2\lambda\boldsymbol{d}^T\tilde{\boldsymbol{\beta}} \tag{6.13}$$

where λ is the Lagrange multiplier.

By taking the partial derivatives of jth coefficient $\tilde{\beta}_j$ of $\tilde{\boldsymbol{\beta}}$:

$$\frac{\partial L(\tilde{\boldsymbol{\beta}})}{\partial\tilde{\beta}_j} = \sum_{i=1}^{m}2(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)\tilde{\boldsymbol{x}}_{2ij} + 2\lambda d_j \tag{6.14}$$

where $\tilde{\boldsymbol{x}}_{2ij}$ is the jth component of $\tilde{\boldsymbol{x}}_{2i}$ and d_j is the jth component of $\boldsymbol{d}$. By setting the

derivative to be zero for all j, we can get:

$$(\sum_{i=1}^{m} \tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{x}}_{2ij})\tilde{\boldsymbol{\beta}} = \sum_{i=1}^{m} y_i\tilde{\boldsymbol{x}}_{2ij} - \lambda d_j \tag{6.15}$$

Thus we can rewrite Equation 6.15 with matrix form:

$$\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2\tilde{\boldsymbol{\beta}} = \tilde{\boldsymbol{X}}_2^T\boldsymbol{y} - \lambda\boldsymbol{d} \tag{6.16}$$

where $\tilde{\boldsymbol{X}}_2$ is the matrix form of $\tilde{\boldsymbol{x}}_{2i}, i \in [m]$ and $\boldsymbol{y}$ is the vector form of $y_i, i \in [m]$. Therefore, we have:

$$\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2)^{-1}(\tilde{\boldsymbol{X}}_2^T\boldsymbol{y} - \lambda\boldsymbol{d}) \tag{6.17}$$

We can also get solution of λ using the fairness constraint $\boldsymbol{d}^T\tilde{\boldsymbol{\beta}} = 0$:

$$\lambda = \frac{\boldsymbol{d}^T(\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2)^{-1}\tilde{\boldsymbol{X}}_2^T\boldsymbol{y}}{\boldsymbol{d}^T(\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2)^{-1}\boldsymbol{d}} \tag{6.18}$$

By substituting λ into Equation 6.17, we have the closed solution. □

Result 2. For fair regression with the mean squared loss and $\text{MSED}(\hat{y}, a) = 0$, we have the closed solution

$$\begin{aligned}\tilde{\boldsymbol{\beta}} =&(\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2 + \frac{\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T\tilde{\boldsymbol{X}}_2^0 - \frac{\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T\tilde{\boldsymbol{X}}_2^1)^{-1} \\ &(\tilde{\boldsymbol{X}}_2^T\boldsymbol{y} + \frac{\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T\boldsymbol{y}_0 - \frac{\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T\boldsymbol{y}_1)\end{aligned} \tag{6.19}$$

Proof. The fair Heckman prediction model can be described as:

$$\begin{aligned}&\min L(\boldsymbol{\beta}) = \sum_{i=1}^{m}(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2 \\ &\text{subject to} \quad \frac{1}{m_0}\sum_{i\in\mathcal{D}_0}(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2 = \frac{1}{m_1}\sum_{i\in\mathcal{D}_1}(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2\end{aligned} \tag{6.20}$$

We use the same notations as above and apply the Lagrange multipliers:

$$L(\tilde{\boldsymbol{\beta}}) = \min \sum_{i=1}^{m} (\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2 + \lambda(\frac{1}{m_0}\sum_{i\in\mathcal{D}_0}(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2 - \frac{1}{m_1}\sum_{i\in\mathcal{D}_1}(\tilde{\boldsymbol{x}}_{2i}\tilde{\boldsymbol{\beta}} - y_i)^2) \tag{6.21}$$

We can compute the derivatives of $\tilde{\boldsymbol{\beta}}$ with the matrix form and set it to be zero:

$$2\tilde{\boldsymbol{X}}_2^T(\tilde{\boldsymbol{X}}_2\tilde{\boldsymbol{\beta}} - \boldsymbol{y}) + \frac{2\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T(\tilde{\boldsymbol{X}}_2^0\tilde{\boldsymbol{\beta}} - \boldsymbol{y}_0) - \frac{2\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T(\tilde{\boldsymbol{X}}_2^1\tilde{\boldsymbol{\beta}} - \boldsymbol{y}_1) = 0 \tag{6.22}$$

where $\tilde{\boldsymbol{X}}_2^0$ is the matrix form of $\tilde{\boldsymbol{x}}_{2i}, i \in [m_0]$, $\tilde{\boldsymbol{X}}_2^1$ is the matrix form of $\tilde{\boldsymbol{x}}_{2i}, i \in [m_1]$, $\boldsymbol{y}_0$ is the vector form of $y_i, i \in [m_0]$, $\boldsymbol{y}_1$ is the vector form of $y_i, i \in [m_1]$, and $\boldsymbol{y}$ is the vector form of $y_i, i \in [m]$. Then we can get:

$$(\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2 + \frac{\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T\tilde{\boldsymbol{X}}_2^0 - \frac{\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T\tilde{\boldsymbol{X}}_2^1)\tilde{\boldsymbol{\beta}} = \tilde{\boldsymbol{X}}_2^T\boldsymbol{y} + \frac{\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T\boldsymbol{y}_0 - \frac{\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T\boldsymbol{y}_1 \tag{6.23}$$

Therefore, the solution of $\tilde{\boldsymbol{\beta}}$ is:

$$\tilde{\boldsymbol{\beta}} = (\tilde{\boldsymbol{X}}_2^T\tilde{\boldsymbol{X}}_2 + \frac{\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T\tilde{\boldsymbol{X}}_2^0 - \frac{\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T\tilde{\boldsymbol{X}}_2^1)^{-1}(\tilde{\boldsymbol{X}}_2^T\boldsymbol{y} + \frac{\lambda}{m_0}(\tilde{\boldsymbol{X}}_2^0)^T\boldsymbol{y}_0 - \frac{\lambda}{m_1}(\tilde{\boldsymbol{X}}_2^1)^T\boldsymbol{y}_1) \tag{6.24}$$

We can then substitute Equation 6.24 to $\text{MSED}(\hat{y}, a) = 0$ and compute the solution of λ and $\tilde{\boldsymbol{\beta}}$. □

6.3.3.3 Duality Gap Analysis

The optimal value d^* of the Lagrange dual problem, by definition, is the best lower bound on p^* that can be obtained from the Lagrange dual function. The difference $p^* - d^*$, which is always nonnegative, is the optimal duality gap of the original problem. One

theoretical question is whether and under what conditions we can achieve zero duality gap (i.e., the optimal values of the primal and dual problems are equal) in our fair regression framework.

Result 3. For fair regression with the convex loss function and the fairness inequality constraints (i.e., less than a user-specified threshold τ), the strong duality holds for Pearson correlation if the linear relationship exists between x and a.

Proof. Our proof is based on strong duality via Slater condition. The Slater condition states that if a convex optimization problem has a feasible point $\tilde{\boldsymbol{\beta}}_0$ in the relative interior of the problem domain and every inequality constraint $g_i(\tilde{\boldsymbol{\beta}}) \leq 0$ is strict at $\tilde{\boldsymbol{\beta}}_0$, i.e., $g_i(\tilde{\boldsymbol{\beta}}_0) < 0$, then strong duality holds. Both MD and MSED can be viewed as a special case of a general set of linear inequalities on conditional moments [111] and satisfy Slater condition. Next we present the proof that Slater condition also holds for Pearson coefficient as fairness constraint.

The correlation usually exists between $\tilde{\boldsymbol{x}}_2$ and a, and then the prediction based on $\tilde{\boldsymbol{x}}_2$ has disparate impact. We can remove the correlation between $\tilde{\boldsymbol{x}}_2$ and a through the following regression:

$$\hat{\boldsymbol{B}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\tilde{\boldsymbol{X}}_2, \boldsymbol{U} = \tilde{\boldsymbol{X}}_2 - \hat{\boldsymbol{B}}\boldsymbol{A} \tag{6.25}$$

where $\boldsymbol{A} = (a_1, a_2, \cdots, a_n)$ and we define $\boldsymbol{u}_i$ as the i-th datapoint of $\boldsymbol{U}$. With the assumption that the linear relationship exists between $\tilde{\boldsymbol{x}}_2$ and a, it was proved by [97] that ($\boldsymbol{u}$, a) has the same information with ($\tilde{\boldsymbol{x}}_2$, a) and the correlation between $\boldsymbol{u}$ and a is $O(\frac{1}{\sqrt{n}})$.

Suppose the prediction outcome $\hat{y}$ is expressed as the following:

$$\hat{y} = a\tilde{\boldsymbol{\beta}}_a + \boldsymbol{u}\tilde{\boldsymbol{\beta}}_u \tag{6.26}$$

Then we can compute the Pearson coefficient as the following:

$$\rho(\hat{y}, a) = \frac{Cov(\hat{y}, a)}{\sqrt{Var(\hat{y})Var(a)}} \tag{6.27}$$

where $Cov(\hat{y}, a)$ is the correlation between $\hat{y}$ and a, $Var(\hat{y})$ is the variance of $\hat{y}$, and $Var(a)$ is the variance of a. $Cov(\hat{y}, a)$ is calculated as:

$$\begin{aligned} Cov(\hat{y}, a) &= Cov(a\tilde{\boldsymbol{\beta}}_a + \boldsymbol{u}\tilde{\boldsymbol{\beta}}_u, a) \\ &= Cov(a\tilde{\boldsymbol{\beta}}_a, a) + Cov(\boldsymbol{u}\tilde{\boldsymbol{\beta}}_u, a) \\ &= \tilde{\boldsymbol{\beta}}_a Var(a) + 0 \\ &= \tilde{\boldsymbol{\beta}}_a Var(a) \end{aligned} \tag{6.28}$$

The variance of $\hat{y}$ is computed as:

$$\begin{aligned} Var(\hat{y}) &= Var(a\tilde{\boldsymbol{\beta}}_a + \boldsymbol{u}\tilde{\boldsymbol{\beta}}_u) \\ &= Var(a\tilde{\boldsymbol{\beta}}_a) + Var(\boldsymbol{u}\tilde{\boldsymbol{\beta}}_u) \\ &= \tilde{\boldsymbol{\beta}}_a^2 Var(a) + \tilde{\boldsymbol{\beta}}_u^T \boldsymbol{V}_u \tilde{\boldsymbol{\beta}}_u \end{aligned} \tag{6.29}$$

where $\boldsymbol{V}_u$ is the covariances of $\boldsymbol{u}$. Thus Equation 6.27 can be written as:

$$\begin{aligned} \rho(\hat{y}, a) &= \frac{Cov(\hat{y}, a)}{\sqrt{Var(\hat{y})Var(a)}} \\ &= \frac{\tilde{\boldsymbol{\beta}}_a Var(a)}{\sqrt{(\tilde{\boldsymbol{\beta}}_a^2 Var(a) + \tilde{\boldsymbol{\beta}}_u^T \boldsymbol{V}_u \tilde{\boldsymbol{\beta}}_u) Var(a)}} \\ &= \frac{\tilde{\boldsymbol{\beta}}_a \sqrt{Var(a)}}{\sqrt{\tilde{\boldsymbol{\beta}}_a^2 Var(a) + \tilde{\boldsymbol{\beta}}_u^T \boldsymbol{V}_u \tilde{\boldsymbol{\beta}}_u}} \end{aligned} \tag{6.30}$$

Up to now, we can write down the fairness regression subject to the fairness constraint of

Pearson coefficient:

$$\begin{aligned} &\min_{\tilde{\boldsymbol{\beta}}} L(\tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{m} l[(f_h(\tilde{\boldsymbol{x}}_{2i}; \tilde{\boldsymbol{\beta}}), y)] \\ &\text{subject to } \rho^2(\hat{y}, a) \leq \epsilon \end{aligned} \tag{6.31}$$

where ϵ is the threshold of the fairness metric. The fairness constraint $\rho^2(\hat{y}, a) \leq \epsilon$ is equivalent to:

$$(1-\epsilon)\tilde{\boldsymbol{\beta}}_a^2 Var(a) - \epsilon \tilde{\boldsymbol{\beta}}_u^T \boldsymbol{V}_u \tilde{\boldsymbol{\beta}}_u \leq 0 \tag{6.32}$$

The Slater condition requires that $\{(\tilde{\boldsymbol{\beta}}_a, \tilde{\boldsymbol{\beta}}_u) : (1-\epsilon)\tilde{\boldsymbol{\beta}}_a^2 Var(a) - \epsilon \tilde{\boldsymbol{\beta}}_u^T \boldsymbol{V}_u \tilde{\boldsymbol{\beta}}_u < 0\} \neq \emptyset$. It can be easily verified that Slater condition holds. For example, we can set $\tilde{\boldsymbol{\beta}}_a$ to be zero. Since $\boldsymbol{V}_u$ is symmetry and we can apply diagonal decomposition for $\boldsymbol{V}_u$ and the eigenvalues of $\boldsymbol{V}_u$ cannot be all zero. Suppose the jth eigenvalue of $\boldsymbol{V}_u$ is non-zero, and we can set the corresponding jth component of $\tilde{\boldsymbol{\beta}}_u$ to be same sign with the jth eigenvalue, and set all other components of $\tilde{\boldsymbol{\beta}}_u$ to be zero, so that the the Slater condition holds. □

Remarks Note that we do not need to conduct the duality gap analysis for the mean difference (MD) and the mean squared error difference (MSED) as Results 1 and 2 have already given the explicit formulas for the primal optimization. For Partial, SP, BGL and other potential fairness notions, we leave their analysis in our future work. Moreover, when there is no sample selection bias, our Results 1-3 naturally hold by removing the tilde from those tilde symbols (e.g., $\tilde{\boldsymbol{\beta}}$).

6.4 Experiments

6.4.1 Experiment Setting

Datasets. We conduct our experiment on three real-world datasets that are widely used to evaluate fair machine learning models. For each dataset, we choose 70% of data as training data $\mathcal{D}$ and leave the rest as testing data. To create the sample selection bias, we follow the procedure in [93] by splitting $\mathcal{D}$ into $\mathcal{D}_s$ (samples with fully observed features X and target Y) and $\mathcal{D}_u$ (samples with missing Y) according to some specific features. We show the characteristics of three datasets including protected attribute A, target Y, sizes of $\mathcal{D}_s$, $\mathcal{D}_u$, and testing data in Table 6.2 and show the attribute lists used in selection/prediction in Table 6.3.

Table 6.2: Characteristics of datasets

Dataset	Protected Attribute A	Target Y	$\lvert\mathcal{D}_s\rvert$	$\lvert\mathcal{D}_u\rvert$	Testing
CRIME	AAPR	Crime Rate	976	419	599
LAW	Black/Non-black	GPA	1373	567	810
COMPAS	Black/Non-black	Risk Score	2153	924	1320

CRIME dataset [113] was collected from the 1990 US Census and contains socio-economic data of 1994 communities. The task is to predict the crime rate of a given community based on its socio-economic information. We choose the African American Population Ratio (AAPR) as the sensitive attribute and label a community as protected if its AAPR is greater than 50% and non-protected otherwise. In total, we have 219 protected communities and 1775 non-protected communities. In our experiments, we remove attributes with missing values and standardize all attributes to have zero mean value and unit variance. We include samples to $\mathcal{D}_s$ if the ratio of people under the poverty level in a community is less than 0.05,

Table 6.3: Attributes used for selection/prediction. Those with italic font are for prediction and those with either regular or italic font are for selection.

Dataset	Attribute
CRIME	population, householdsize, racepctblack, racePctWhite, racePctAsian, *racePctHisp*, *agePct12t21*, *agePct12t29*, *agePct16t24*, *agePct65up* *numbUrban*, *pctUrban*, *medIncome*, *pctWWage*, *pctWFarmSelf*, *pctWInvInc*, *pctWSocSec*, *pctWPubAsst*, *pctWRetire*, *medFamInc*
LAW	cluster, lsat, ugpa, zgpa, *fulltime*, *fam_inc*, *age*, *gender*, *pass*
COMPAS	decile_score.1, age_cat_25 - 45, age_cat_Greater than 45, age_cat_Less than 25, c_charge_degree_F, c_charge_degree_M, *sex*, *age*, *juv_fel_count*, *juv_misd_count*, *juv_other_count*, *priors_count*, *two_year_recid*

and samples to $\mathcal{D}_u$ otherwise.

LAW dataset [114] was collected from the Law School Admissions Council's National Longitudinal Bar Passage Study and consists of personal records of law students who went on to take the bar exam, including LSAT score, age, race and so forth. The task is to predict the GPA of a student based on other attributes. We choose race as the sensitive attribute and treat black as protected. The dataset contains a total of 20649 records and we randomly select 3000 records, including 500 protected samples and 2500 non-protected samples. We include samples to $\mathcal{D}_s$ if the year of birth is after 1950, and samples to $\mathcal{D}_u$ otherwise.

COMPAS dataset [7] consists of a collection of data from criminal defenders from Florida in 2013-2014. Each data sample is associated with personal information, including race, gender, age, prior criminal history, and so forth. The task is to predict the risk level of a defender based on other attributes. We choose race as the sensitive attribute and treat black defenders as protected. After removing the duplicated data samples, we have a total of 4397 data samples, including 2694 protected samples and 1703 non-protected samples. We

include samples to $\mathcal{D}_s$ if the year of decile score is less than 10, and to $\mathcal{D}_u$ otherwise.

Baseline Models and Metrics. We choose linear regression with the standard loss function, mean squared loss, in our proposed framework *FairLR**. We adopt each of four fairness metrics, MD, MSED, Pearson coefficient and Partial coefficient, with equality constraint forms. We consider the following baseline models: (a) Linear regression (*LR*) without fairness constraint; (b) Linear regression with Heckman correction (*Heckman*) from [21] (c) Linear regression with each fairness constraint (*FairLR*), including MD [95], MSED [96], Pearson coefficient [97], and Partial coefficient. We evaluate the performance of the proposed framework based on prediction accuracy and fairness. We use the mean squared error (MSE) to measure prediction accuracy. For fairness, we use MD and MSED in the binary sensitive attribute setting and Pearson coefficient and Partial coefficient in the numerical sensitive attribute setting. As the goal of fair regression is to achieve good accuracy and fairness on population, we use MSE and fairness calculated from testing data to compare different models. For a comprehensive comparison, we also report those values calculated from $\mathcal{D}_s$.

6.4.2 Performance Evaluation on Binary Protected Attribute

We report in Figure 6.2 our main comparison results on three datasets. Y-axis is MSE to reflect prediction accuracy and X-axis is based on the fairness metric chosen in fair regression models (*FairLR* and *FairLR**). In particular, the three plots in the first row of Figure 6.2 report MD whereas those on the second row report MSED. In each plot, we have eight markers with different shape and color, each of which reflects the MSE and fairness metric for one of the four compared models on either $\mathcal{D}_s$ or testing data. Throughout this section, we use $\circ$, $\triangle$, $\diamond$, and $\star$ to denote *LR*, *Heckman*, *FairLR* and *FairLR**, and use hollow

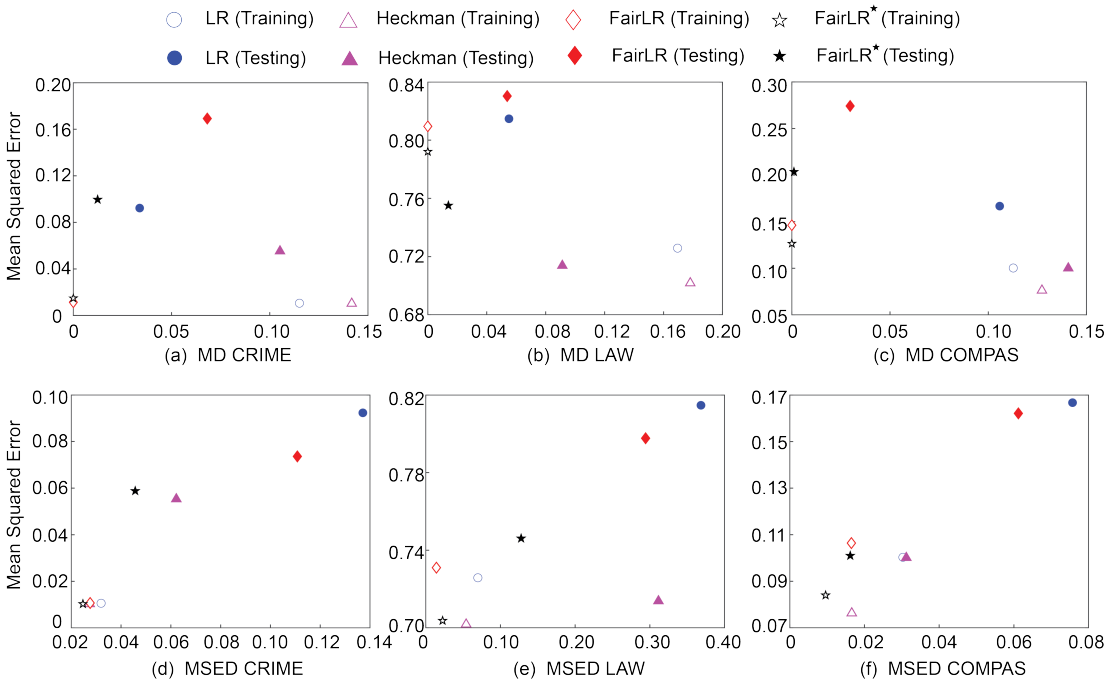

Figure 6.2: Performance evaluation of binary protected attribute on CRIME, LAW, and COMPAS

(solid) marker to represent results on $\mathcal{D}_s$ (testing data). In general, markers in bottom-left region (close to origin) indicate good performance of corresponding methods as we want to achieve both low MSE for prediction and low MD/MSED for fairness.

We focus on the main results of comparing four methods on testing data, reflected by four solid markers in each plot. We clearly see that markers of *Heckman* always locate below that of *LR*, indicating *Heckman* successfully corrects the sample selection bias and reduces prediction error on testing data. Taking Figure 6.2 (a) as an example, the MSE of *Heckman* is 0.0553 whereas the MSE of *LR* is 0.0923. Similarly, as *FairLR* does not do bias correction, the solid marker of *FairLR* is also higher than that of *FairLR** for all three datasets. This demonstrates the effectiveness of Heckman model for correcting sample selection bias. Moreover, the solid marker of *FairLR* is always on the right side of *FairLR**, reflecting that *FairLR* simply trained on $\mathcal{D}_s$ without bias correction fails to achieve fairness

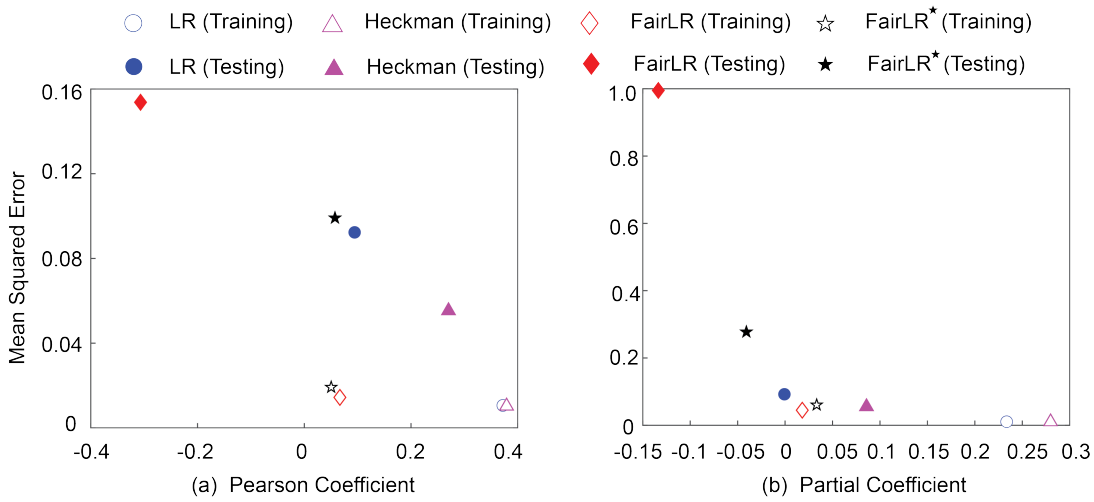

Figure 6.3: Performance evaluation of numerical protected attribute on CRIME

on testing data. For example, in Figure 6.2 (f), the MSED of *FairLR* is 0.0612 whereas that of *FairLR** is 0.0162. To conclude, our proposed *FairLR** achieves the best trade-off between fairness and regression accuracy on the testing data.

It is also interesting to compare each model's performance between the training data and testing data. For our *FairLR**, we can see its hollow marker and solid marker are close to each other horizontally, indicating that the fairness achieved on $\mathcal{D}_s$ can also guarantee the testing fairness. However, the hollow marker and solid marker of *FairLR* are separate, indicating the sample selection bias can incur unfairness in the testing data although *FairLR* achieved training fairness.

6.4.3 Performance Evaluation on Numerical Protected Attribute

We conduct experiments on CRIME by using the original numerical attribute AAPR as sensitive attribute. We use Pearson coefficient to measure the independence between $\hat{Y}$ and A, and use Partial coefficient to measure the conditional independence of $\hat{Y}$ and A given the true target value Y. Figure 6.3 shows the comparison results of four models. We

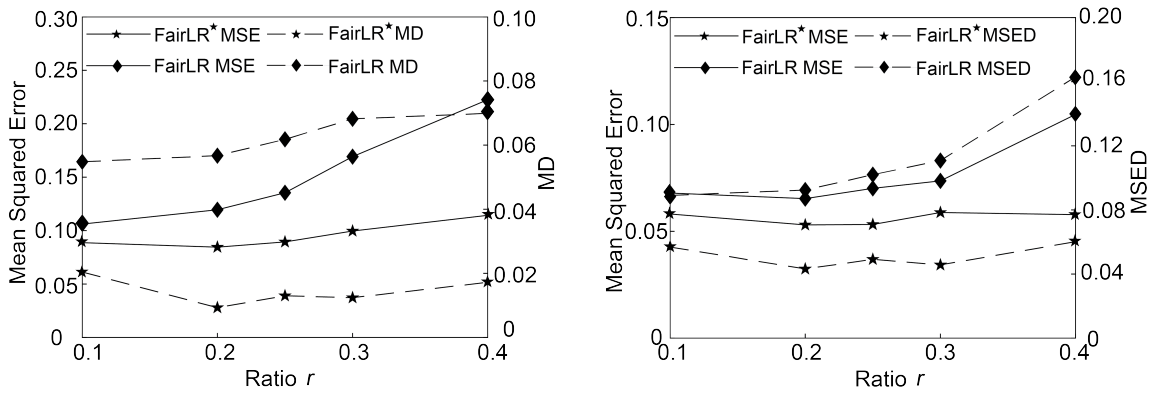

Figure 6.4: Effects of Ratio $r = |\mathcal{D}_u|/|\mathcal{D}|$

have similar observations as the binary protected attribute setting. First, for results on $\mathcal{D}_s$ reflected by hollow markers, there is no surprise that all of the hollow markers are in bottom region with low MSE, and both *FairLR* and *FairLR** can achieve training fairness in terms of both Pearson coefficient and Partial coefficient. Second, for results on testing data reflected by solid markers, *Heckman* (*FairLR**) achieves lower MSE than *LR* (*FairLR*) as the former model considers the sample bias selection. More importantly, only *FairLR** is able to achieve testing fairness given the fairness threshold.

6.4.4 Performance Evaluation on Biased Ratio

In this section, we evaluate how ratio $r = |\mathcal{D}_u|/|\mathcal{D}|$ would affect the performance of our *FairLR** and baseline *FairLR* on the testing data. Note that larger r indicates more bias in sample selection. We conduct experiments on CRIME. Figure 6.4 plots results of MD and MSED. In both plots, X-axis shows the varied r values from 0.1 to 0.4, the left Y-axis shows MSE, and the right Y-axis shows the fairness metric (MD or MSED). Correspondingly, we use solid lines to represent MSE values and dashed lines to represent fairness values. It is unsurprising to see that *FairLR** always achieves better performance (smaller MSE and smaller MD or MSED) than *FairLR*, as demonstrated in Figure 6.4 that lines with symbol

$\star$ locate below those with symbol $\diamond$. More importantly, for our *FairLR**, the fairness value (MD or MSED) and prediction error (MSE) are stable when r increases, demonstrating the robustness of our *FairLR** against sample selection bias. On the contrary, for *FairLR*, both the unfairness and prediction error on the testing data increase when r increases.

6.5 Summary

In this chapter, we have developed a framework for fair regression under sample selection bias when dependent variable values of a set of samples are missing. The framework adopts the classic Heckman model to correct sample selection bias and captures a variety of fairness notions via inequality and equality constraints. We applied the Lagrange duality theory to derive the dual convex optimization and showed the conditions of achieving strong duality for Pearson correlation. For the two popular fairness notions, mean difference and mean squared error difference, we further derived explicit formulas without optimizing iteratively. Experimental results on three real-world datasets demonstrated our approach's effectiveness.

7 Enhancing Personalized Modeling via Weighted and Adversarial Learning

7.1 Introduction

The past few years have witnessed an increasing role that deep learning plays in various kinds of applications, such as image classification [4], text generation [115], recommendation systems [116] and other artificial intelligence (AI)-related tasks. More recently, data for training deep learning models is increasingly distributed among different users. A traditional approach for the deployment of deep learning model is to collect all these data into a central server and build a global model. An alternative way is to collaboratively learn a global model among distributed users via parameters exchange [64]. The key of previous works is to build a one-fit-all global model and use it to perform classification or prediction for all users. Although a global model captures generic information of training data from all users, it cannot fulfill the personalized demand for each individual user due to the diverse data distribution among different users.

It is necessary to construct a personalized model for each individual user to improve intelligent services. Although the global model learns generic information over all users, the specific features of each individual user's data, which are of vital importance to build the personalized data, are often overlooked by the global model. In many circumstances, building a personalized model can achieve better performance for specific users. For instance, personalized recommendation systems play important roles for many online services [117], such as production advertisement, sales promotion, and information recommendation. In addition, electronic health records [118] are used to predict disease progression and provide treatment plans. Since electronic health records are patient-specific, personalized models are

more effective for individual users.

In fact, in many cases, an individual user can only collect a small amount of data, which is insufficient to build an accurate personalized model to achieve satisfactory performance. To improve the performance of personalized model, an individual user needs to request similar or complementary data from other users. Pioneer works of building personalized model have been done in medical field, e.g., Cheng et al. [118] develop a framework of collecting data from similar patients and then training a personalized model on the combination of these collected data and his own data.

However, there are two challenges in the pipeline of building personalized models. The first challenge is how to efficiently and effectively collect similar data from other users. It is obvious that randomly selecting data from other users may not improve the performance of personalized model as those collected data may be different from the user's own data. Previous research [118] applies similarity metrics to construct similarity index for data selection. However, it is burdensome to construct similarity index and the similarity metrics are often based on manually designed features. The computational cost is high when one compares paired data one by one due to the high operation complexity of $\mathcal{O}(n^2)$ especially when the number of data n is large. Moreover, it is often unclear to choose one appropriate similarity metric so most appropriate data can be collected. Another challenge is how to better train the personalized model with the requested data. Previous research simply combines the collected data with user's own data, which may not build the personalized model with high accuracy. This is because there may exist distribution discrepancy between the collected data and user's own data, especially when the user can only get a small amount of data from other users (e.g., due to privacy concerns or high data collection cost). Hence it is imperative to develop personalized learning models that can handle the potential distribution discrepancy.

Motivated by the above two challenges, in this chapter, we develop a learning framework that enables an individual user to effectively collect data and build a robust model on the combination of the collected data and his own data. For data collection, we propose an approach of using an auto-encoder and a generative adversarial network (GAN) [119]. The auto-encoder is used to obtain data representation that is further used to train the GAN. The trained encoder and the discriminator from GAN are sent to his neighbors. Each neighbor user uses the encoder to obtain the representation of his data and then uses the discriminator to calculate the probability score of his data. Data with high probability scores are combined with the user's original data to train the personalized model. The advantage of using GAN is that it can capture inherent properties of the underlying data without manually specifying features. With the requested data, we develop two approaches to improve the performance of the personalized model. The first approach is weighted learning by assigning a different weight to each record of the requested data. The data record with a high probability score computed by the discriminator is assigned with a high weight. The weighted learning is able to capture the importance of different data. The second approach is the adversarial learning that aims to minimize the distribution discrepancy between the requested data and user's own data. The core idea of the adversarial learning is to map both the requested data and user's own data into the same feature space where the distribution discrepancy is minimized. Our adversarial training is analogous to the discriminator of the GAN. The role of the discriminator is to predict whether the generated features are from user's own data or the requested data.

The main contributions of this chapter are summarized as follows. First, we demonstrate that building a global model is not an optimal choice for personalized prediction. Second, we develop a strategy based on auto-encoder and GAN to effectively collect simi-

lar data from other users. Third, instead of directly using the requested data for training, we design two approaches, weighted learning and adversarial learning, to further improve the performance of the personalized model. Finally, we conduct extensive experiments and the results demonstrate the efficiency of the proposed framework. Note that, this chapter is originally from the published works [120, 121].

7.2 Related Work

Personalized deep learning and recommendation: Personalized model is popular in the areas of medicine and recommendation systems. In these areas, building a personalized model for each individual user is necessary since medical data and recommendation service are user-specific and a global model cannot capture personalized features. In personalized medical prediction area, similarity learning is fundamental for building a personalized patient model. In [122], Che et al. propose a dynamic temporal matching approach to find similar data for individual users and build a personalized RNN model for each individual user to predict disease. The authors [123] develop a CNN-based similarity method for paired data comparison and perform personalized disease prediction. Other works using different metrics for similarity personalized learning [124, 125, 126] are also reported.

Personalized recommendation has also made rapid progress in many areas, such as e-commerce, advertising, audio and video recommendation [127]. Collaborative filtering, which is one of the most popular recommendation approaches, makes predictions about the interests of a user by collecting preferences from many similar users. Various metrics are used to measure the similarity between different users. For instance, Luo et al. [128] apply cosine similarity to build recommendation system for smart grid end users and Kouki et al. [129] use Pearson's correlation to compute data similarity. Recently, deep learning models are widely

used to improve the performance of personalized recommendation. For example, Hu et al. [130] propose to diversify personalized recommendation results by leveraging user-session contexts and designing session-based neural networks to efficiently learn session profiles over large number of users and items. Yu et al. [131] design an attention-based LSTM framework to generate users' representations and adaptively to learn and predict according to the specific context.

Different from above works, we propose an efficient similarity approach based on auto-encoder and GAN and apply adversarial training to reduce potential distribution discrepancy. In fact, our work can be integrated into personalized recommendation systems to improve their performance. For example, our auto-encoder and GAN based similarity approach can be used to efficiently group different users in collaborative filtering recommendation. Moreover, the adversarial training can also be applied to learn and differentiate the representations of different users and then improve the personalized prediction in content-specific scenarios.

Representation learning: Representation learning has been a well studied research area in the past few years [132], especially in computer vision, natural language processing and transfer learning. There are several works that apply representation learning to solve distribution discrepancy between different parties. For example, Tzeng et al. propose to learn domain invariant representations from the source domain and transfer to the target domain for prediction [133]. Liu et al. develop a framework that disentangles domain-invariant and domain-specific features in image translation and manipulation [134]. In [135], Gupta et al. extract invariant features between different agents in the reinforcement learning. Misra et al. propose a multi-task model to learn common representations between different tasks and use it to improve prediction performance [136]. Our work falls into the general area that exploits feature representation among different groups for performance improvement [137, 138]. Dif-

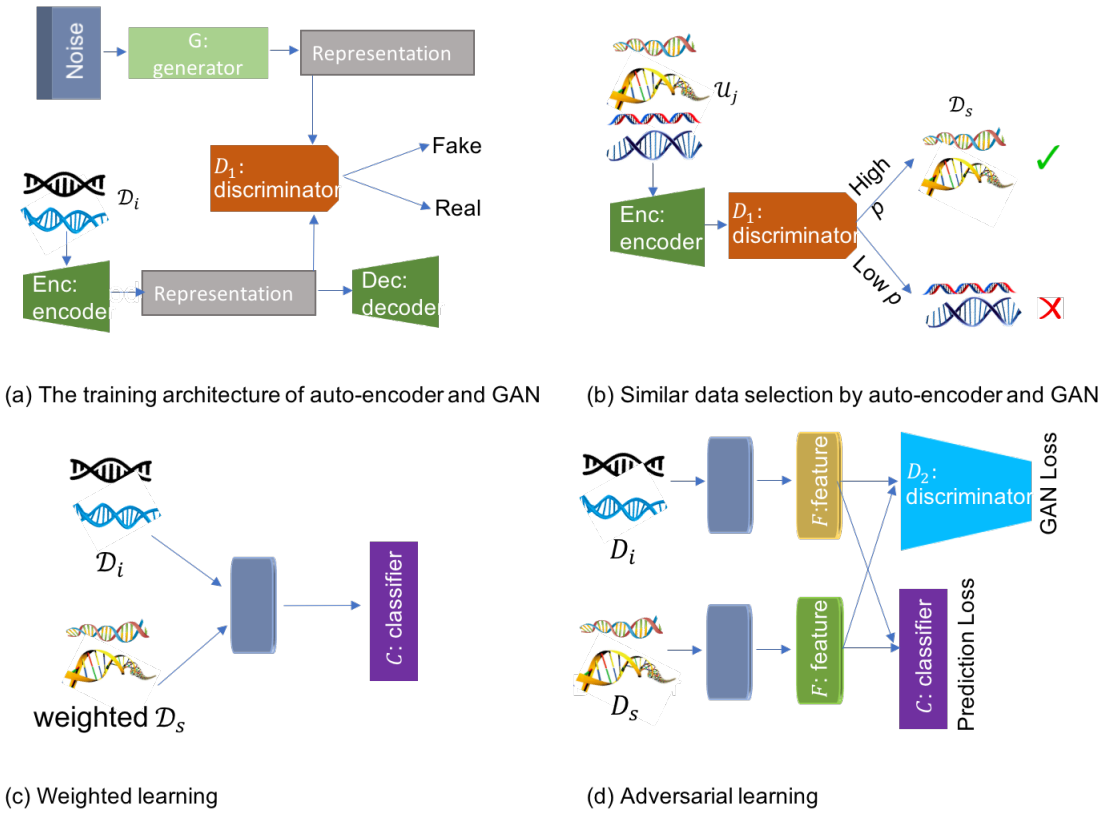

Figure 7.1: The framework of personalized learning where (a), (b) and (c) is for weighted learning and (a), (b) and (d) is for adversarial learning.

ferent from previous works, our work applies adversarial training to reduce the distribution discrepancy and learns the prediction task simultaneously to improve its performance.

7.3 Weighted Learning and Adversarial Learning

7.3.1 Framework Overview

Consider there exist N individual users in this distributed setting and each user has his own local dataset that contains features X and label Y. Let $\mathcal{U}$ and $\mathcal{D}$ represent the set of users and datasets, respectively, where each user $\mathcal{U}_i$ is associated with dataset $\mathcal{D}_i$. Suppose $\mathcal{D}_i$ contains M_i data samples, namely $(X_1, Y_1), (X_2, Y_2)\ , \ldots, (X_{M_i}, Y_{M_i})$. The goal of each user $\mathcal{U}_i$ is to build a personalized classification model f_i which takes $\mathcal{D}_i$ as input and minimizes

the prediction loss $\mathcal{L}_i = \sum_{m=1}^{M_i} l_i(f_i(X_m), Y_m)$, where l_i is the given loss function. To boost the performance of classifier f_i and reduce prediction loss $\mathcal{L}_i$, user $\mathcal{U}_i$ will request similar data from his neighbors. The neighbors can be defined from different ways. For example, in a sensor network, the neighbors can be sensors within a physical region. In our experiments, we treat the neighbors of $\mathcal{U}_i$ as all other users except $\mathcal{U}_i$.

For the personalized learning, the joint distribution of $\mathcal{U}_i$ could be different from that of its neighbour $\mathcal{U}_j$, i.e., $P(X_{\mathcal{U}_i}, Y_{\mathcal{U}_i}) \neq P(X_{\mathcal{U}_j}, Y_{\mathcal{U}_j})$. The most rigorous assumption is that $P(X_{\mathcal{U}_i}) \neq P(X_{\mathcal{U}_j})$ and $P(Y_{\mathcal{U}_i}|X_{\mathcal{U}_i}) \neq P(Y_{\mathcal{U}_j}|X_{\mathcal{U}_j})$. Although the overall distribution of $\mathcal{D}_i$ and $\mathcal{D}_j$ are not the same, it is possible that a subset of $\mathcal{D}_j$ at user $\mathcal{U}_j$ could be similar to $\mathcal{D}_i$ and hence in our framework $\mathcal{U}_i$ can request a subset of similar data, $\mathcal{D}_s$, from its neighbours such that $P(X_{\mathcal{U}_i}) \approx P(X_{\mathcal{U}_s})$ and thus $P(Y_{\mathcal{U}_i}|X_{\mathcal{U}_i}) \approx P(Y_{\mathcal{U}_s}|X_{\mathcal{U}_s})$.

However, there could still exist possible distribution shift between $\mathcal{D}_i$ and $\mathcal{D}_s$, therefore, the second part of our framework is focused on how to better use the requested data to build a personalized model. We propose two approaches on how to combine the requested data $\mathcal{D}_s$ with $\mathcal{D}_i$ to build a personalized model for $\mathcal{U}_i$. The first approach is weighted learning shown as Figure 7.1(a), (b) and (c), while the the second approach is adversarial learning shown as Figure 7.1(a), (b) and (d). We show the workflow of our personalized learning framework in Algorithm 3. The workflow has three subsections.

Train auto-encoder and GAN: The first subsection (lines 2 - 4) is to train auto-encoder and GAN for $\mathcal{U}_i$ based on his own dataset $\mathcal{D}_i$. Line 3 trains a normal auto-encoder for $\mathcal{U}_i$. With the trained auto-encoder, we apply its *Enc* to compute the representations of $\mathcal{D}_i$ and use the learned representations as the input to train the GAN. For clear illustration, we show the training process in Figure 7.1(a).

Request $\mathcal{D}_s$ for $\mathcal{U}_i$: The second subsection (lines 6 - 8) is to request similar data for $\mathcal{U}_i$

from his neighbors. This subsection is also illustrated in Figure 7.1(b). $\mathcal{U}_i$ first sends his Enc and D_1 to his neighbors. Each neighbor $\mathcal{U}_j$ uses Enc to compute the representation of each sample and then uses discriminator D_1 to obtain a probability score p for each sample. $\mathcal{U}_j$ collects all data samples with score $p > \tau$, puts into $\mathcal{D}_s$, and sends $\mathcal{D}_s$ back to $\mathcal{U}_i$.

Personalized learning: The final subsection is to build a personalized model for $\mathcal{U}_i$ based on $\mathcal{D}_i$ and $\mathcal{D}_s$. We propose two approaches to train a personalized model. The first approach is the weighted learning (Figure 7.1(c)) and lines 10-14 show its training process. $\mathcal{U}_i$ first uses the discriminator D_1 to compute the probability score p of each data sample from $\mathcal{D}_s$ and uses p as the weight for each data from Equation 7.7. With the probability score, the personalized model can better capture the importance of each requested data sample. The second approach is the adversarial training (Figure 7.1(d)) and lines 15 -22 show its training process. The adversarial training can reduce the distribution discrepancy between $\mathcal{D}_i$ and $\mathcal{D}_s$, which can further improve the performance of f_i compared to training f_i directly over $\mathcal{D}_i$ and $\mathcal{D}_s$.

In fact, the proposed personalized learning is a general framework. It can apply to different machine learning models, such as logistic regression, convolutional neural networks, and long short-term memory neural networks. In our framework, the goal of the auto-encoder and GAN is used to select similar data from other users. In fact, we do not need to train a perfect GAN using lots of data as previous research on generating fake images/sentences to fool human. When the user has limited data, the auto-encoder and GAN trained with insufficient data may select data samples that are not similar to the individual's own data. This is the reason why we propose AdvPL to further reduce the distribution discrepancy between the individual user's data and requested data.

7.3.2 Train Auto-Encoder and GAN for Individual User

The first step of AdvPL is to train the auto-encoder and GAN for $\mathcal{U}_i$. Typically both *Enc* and *Dec* are deep neural networks. The *Enc* takes the data in $\mathcal{D}_i$ and obtains its hidden representation. The hidden representation is then fed into the *Dec* to output a reconstructed input. The advantage of using representations by the auto-encoder is to make the training of the GAN easier, especially for sequential data. The *Enc* obtains hidden representation of data in $\mathcal{D}_i$ and GAN takes it as input.

GAN has a very appealing property: its discriminator can implicitly learn hidden similarity metric and use it to discriminate real data and fake data. As a result, the discriminator is able to capture data distribution pattern of $\mathcal{U}_i$.

Encoder: The encoder is a neural network which takes mth data X_m and outputs a hidden representation as:

$$\mathbf{h}_m^{Enc} = Enc(X_m), \tag{7.1}$$

where $\mathbf{h}_m^{Enc}$ is the hidden representation of X_m and $Enc(\cdot)$ denotes the computation of hidden representation by the encoder neural network. The hidden representation is deemed to capture the information of the input data, which will be the input of the decoder.

Decoder: The decoder is used to reconstruct original input based on $\mathbf{h}_m^{Enc}$ and the reconstruction of X_m is expressed as:

$$X'_m = Dec(\mathbf{h}_m^{Enc}), \tag{7.2}$$

where X'_m is the reconstructed data of X_m and $Dec(\cdot)$ denotes the computation of reconstructed input by the decoder neural network.

Algorithm 3 The framework of personalized learning

Input: $\mathcal{U}_i$ with dataset, $\mathcal{D}_i$ and threshold τ, Option: [Weighted learning, Adversarial learning]
Output: Well trained personalized model f_i;
/* Train auto-encoder and GAN */:
Initialize parameters in auto-encoder (*Enc*, *Dec*) and GAN(G, D_1) for $\mathcal{U}_i$;
$\mathcal{U}_i$ trains the auto-encoder using $\mathcal{D}_i$ and updates parameters for *Enc*, *Dec* using Equation 7.1, 7.2 and Equation 7.3;
$\mathcal{U}_i$ trains the GAN using representations by *Enc* and updates parameters for G, D_1 using Equation 7.4;
/* Request $\mathcal{D}_s$ for $\mathcal{U}_i$ */:
$\mathcal{U}_i$ sends *Enc* and D_1 to his neighbor users $\mathcal{U}_j$;
Requested dataset $\mathcal{D}_s = \emptyset$;
For each neighbor user $\mathcal{U}_j$: encode $\mathcal{D}_j$ by *Enc*, compute probability score p using D_1, and put data in $\mathcal{D}_s$ with $p > \tau$;
If Option = Weighted learning:
    /* Weighted training */:
    while not converged:
        Compute probability score for each data from $\mathcal{D}_s$ from Equation 7.6;
        Compute weighted loss based on $\mathcal{D}_i$ and $\mathcal{D}_s$ from Equation 7.7 and update model parameters;
Elif Option = Adversarial learning:
    /* Adversarial training */:
    Initialize parameters in feature extractor (F), model classifier (C), and discriminator (D_2) for adversarial training;
    $\mathcal{U}_i$ trains C and F to optimal performance using $\mathcal{D}_i$;
    while not converged:
        Fix D_2 and F, and update C with loss Equation 7.10 over $\mathcal{D}_i$ and $\mathcal{D}_s$;
        Fix C, update D_2 and F with loss Equation 7.9 and 7.8 over $\mathcal{D}_i$ and $\mathcal{D}_s$;
return personalized model f_i

The performance of the auto-encoder is evaluated by the distance between X_m and X'_m and the loss over $\mathcal{D}_i$ is expressed as:

$$\mathcal{L}_{AE} = \frac{1}{M_i} \sum_{m=1}^{M_i} (X_m - X'_m)^2 \tag{7.3}$$

After the training process of auto-encoder, $\mathcal{U}_i$ can use the *Enc* to transform input data into hidden representation and train the GAN based on the representation data. The

GAN consists of a generator G and a discriminator D_1. The goal of G is to generate fake representation $\mathbf{h}_{fake}^{Enc}$ and D_1 is to distinguish real representation $\mathbf{h}_{real}^{Enc}$ and fake representation $\mathbf{h}_{fake}^{Enc}$. The objective function of the GAN is the same as the traditional GAN in previous work [119]:

$$\begin{aligned}\min_{G}\max_{D_1} \mathbb{E}_{\mathbf{h}_{real}^{Enc}\sim P_{data}} \log D_1(\mathbf{h}_{real}^{Enc}) \\ + \mathbb{E}_{\mathbf{h}_{fake}^{Enc}\sim P(\mathbf{G})} \log(1 - D_1(G(\mathbf{h}_{fake}^{Enc}))),\end{aligned} \tag{7.4}$$

where P_{data} ($P_{\mathbf{G}}$) denotes the probability distribution of $\mathcal{D}_i$ (noise).

The data distribution of $\mathcal{U}_i$ is captured by his auto-encoder and GAN. To request similar data, $\mathcal{U}_i$ only needs to send Enc and D_1 to his neighbor users $\mathcal{U}_j$. If a data point of $\mathcal{U}_j$ passes the D_1 with high probability score, then this data is more likely to be sampled from the same distribution of $\mathcal{U}_i$ and can be put into $\mathcal{D}_s$.

7.3.3 WL: Weighted Learning for Personalized Model

A straightforward approach is to directly incorporate $\mathcal{D}_s$ to his personalized model so that we have the combined $\mathcal{D}_s$ and $\mathcal{D}_i$ as the training data. The total loss based on the combination of $\mathcal{D}_s$ and $\mathcal{D}_i$ is expressed as:

$$\mathcal{L}_{unweighted} = \sum_{m=1}^{|\mathcal{D}_i|} l_i(f_i(X_m), Y_m) + \sum_{n=1}^{|\mathcal{D}_s|} l_i(f_i(X_n), Y_n), \tag{7.5}$$

where l_i is the loss function and f_i is the classifier for $\mathcal{U}_i$. In Equation 7.5, each data sample in $\mathcal{D}_i$ and $\mathcal{D}_s$ is assigned with the same weight.

However, the importance of data samples in $\mathcal{D}_s$ should not be the same and generally should be less than data samples in $\mathcal{D}_s$ due to the distribution discrepancy. As we discussed,

the neighbor user $\mathcal{U}_j$ uses Enc and D_1 to compute the probability score p for each data sample as the following:

$$p_n = D_1(Enc(X_n)), \tag{7.6}$$

where X_n is the nth data from $\mathcal{D}_s$. As aforementioned, the discriminator D_1 captures the distribution of $\mathcal{D}_i$. Therefore, p_n can be seen as a metric to measure the similarity between X_n and $\mathcal{D}_i$. Higher p_n indicates that X_n is more likely to be sampled from the distribution of $\mathcal{D}_i$. Hence we take p_n into consideration and assign p_n as the weight of X_n to construct a weighted loss:

$$\mathcal{L}_{weighted} = \sum_{m=1}^{|\mathcal{D}_i|} l_i(f_i(X_m), Y_m) + \sum_{n=1}^{|\mathcal{D}_s|} p_n l_i(f_i(X_n), Y_n), \tag{7.7}$$

The intuition of Equation 7.7 is that it assigns a higher (lower) weight for the data sample which is more (less) similar to the distribution of $\mathcal{D}_i$. Consider the following two cases. First, if the probability scores of all requested data from $\mathcal{D}_s$ are approaching to 1, then the weighted loss Equation 7.7 is reduced to the unweighted loss Equation 7.5, indicating that the unweighted loss is a special case of the weighted loss. Second, if the probability scores of the requested data are mixed, then the weighted loss can capture the importance difference of different data samples so that it can better improve the performance of the personalized model.

7.3.4 AdvPL: Adversarial Learning for Personalized Model

We explain the motivation and necessity of the adversarial training. Our aim is to reduce the distribution discrepancy between $\mathcal{D}_i$ and $\mathcal{D}_s$. As shown in Figure 7.2(a), the data in $\mathcal{D}_s$ ($\mathcal{D}_i$) are represented by circular (triangular) dots. In raw data space, $\mathcal{D}_s$ and $\mathcal{D}_i$ are

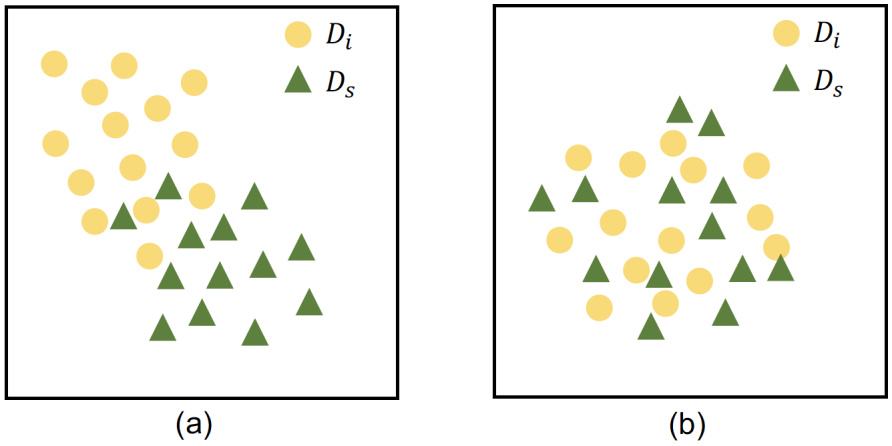

Figure 7.2: (a) It depicts raw data distribution using dots and triangles between $\mathcal{D}_i$ and $\mathcal{D}_s$. In raw data space, the distribution between $\mathcal{D}_i$ and $\mathcal{D}_s$ is partly overlapped. (b) By optimizing a loss function that simultaneously maximizes overlap in the feature representation space and improves the model performance, we can reduce the distribution discrepancy between $\mathcal{D}_s$ and $\mathcal{D}_i$.

partially overlapped and there exists a distribution discrepancy between them. Consequently, the model performance may not be optimal if $\mathcal{U}_i$ directly uses them to train the personalized model. However, if we can transform the raw data into another space where $\mathcal{D}_i$ and $\mathcal{D}_s$ can be well overlapped as shown in Figure 7.2(b), then the distribution discrepancy in this new space will be reduced and these new transformed data can be used to build a more accurate model. To minimize the distribution discrepancy between $\mathcal{D}_s$ and $\mathcal{D}_i$, we present an adversarial learning framework as shown in Figure 7.1(d). The adversarial learning simultaneously reduces the distribution discrepancy between $\mathcal{D}_s$ and $\mathcal{D}_i$ in the feature space and trains the model with these feature data simultaneously.

The adversarial training for personalized model is composed of three parts including feature representation extractor (F), discriminator D_2 and model classifier C. It should be mentioned that F and C are two parts of a complete neural network model, so the extracted representations by F can be directly used as intermediate input for C. F is used to extract representations of data in $\mathcal{D}_i$ and $\mathcal{D}_s$. The adversarial training process is analogous to the

traditional GAN. The role of F is to mimic the function of the generator in GAN while the role of the discriminator D_2 is to distinguish the feature representations between $\mathcal{D}_i$ and $\mathcal{D}_s$ created by F. In our proposed AdvPL, F is trained in a manner that maps the data in $\mathcal{D}_i$ and $\mathcal{D}_s$ to feature representations with binary labels, where the label is 1 if feature representation belongs to $\mathcal{D}_i$ and is 0, otherwise. The key of the AdvPL is that F and D_2 are trained together through the adversarial learning process. More specifically, the goal of F is to generate the feature representations of data in $\mathcal{D}_i$ and $\mathcal{D}_s$ into the same space. F is to fool D_2 and makes D_2 unable to distinguish the representations between $\mathcal{D}_i$ and $\mathcal{D}_s$. On the contrary, D_2 is trained to predict whether the feature representations are from $\mathcal{D}_i$ or $\mathcal{D}_s$.

The loss function of F is similar to the generator in traditional GAN and has the following expression:

$$\mathcal{L}_F = -\mathbb{E}_{Z\sim\mathcal{D}_s} \log D_2(F(Z)) \tag{7.8}$$

where $F(Z)$ is the representation of data in $\mathcal{D}_s$.

The discriminator D_2 is trained to identify whether a data sample is from $\mathcal{D}_i$ or $\mathcal{D}_s$ given the feature representations. It is obvious that if the transformed representation suffers from huge distinction between $\mathcal{D}_i$ and $\mathcal{D}_s$, then D_2 is able to separate them easily. The adversarial loss of D_2 is:

$$\mathcal{L}_{D_2} = -\mathbb{E}_{X\sim\mathcal{D}_i} \log D_2(F(X)) - \mathbb{E}_{Z\sim\mathcal{D}_s} \log(1 - D_2(F(Z))) \tag{7.9}$$

where $F(X)$ denotes the representations of his own data in $\mathcal{D}_i$. It can be seen that D_2 plays the same role as the traditional discriminator in traditional GAN.

The combination of $\mathcal{L}_F$ and $\mathcal{L}_{D_2}$ can mimic the adversarial training process of traditional GAN. Through the adversarial learning process, the data in $\mathcal{D}_i$ and $\mathcal{D}_s$ are transformed

into the hidden representation space. In this space, D_2 cannot tell the difference between them so that the distribution discrepancy between $\mathcal{D}_i$ and $\mathcal{D}_s$ can be minimized.

Different from the traditional GAN which only generates realistic examples, our ultimate goal is to train a more accurate model for individual user. As shown in Figure 7.1(d), the extracted representations from $\mathcal{D}_i$ and $\mathcal{D}_s$ will be fed into C and the loss function of C with K classes is:

$$\mathcal{L}_C = -\sum_{m=1}^{M_i}\sum_{k=1}^{K} \mathbb{1}_{k=Y_m} \log C(F(X_m)) - \sum_{m=1}^{|\mathcal{D}_s|}\sum_{k=1}^{K} \mathbb{1}_{k=Y_m} \log C(F(Z_m)), \tag{7.10}$$

where C is the final layer of model classifier and Y_m are the corresponding labels. M_i is the data number of $\mathcal{D}_i$ and $|\mathcal{D}_s|$ is the size of $\mathcal{D}_s$. Therefore, the full framework is to minimize the joint loss function $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_C + \mathcal{L}_F + \mathcal{L}_{D_2}, \tag{7.11}$$

where $\mathcal{L}$ denotes the sum loss of the adversarial learning framework. The joint training process of the adversarial learning framework is summarized in Lines 17 - 21 in Algorithm 1. It mainly includes two parts. The first part is that $\mathcal{U}_i$ trains C and F using $\mathcal{D}_i$. Our goal is to build a personalized model and use similar data $\mathcal{D}_s$ to improve model performance, so we first achieve optimal model performance based on $\mathcal{D}_i$. The second step is to alternatively train C, F and D_2. To train C, we fix F and D_2. To train F and D_2, we fix C. In this way, the balance of the adversarial training will be under better control. To be noted here, we first train the personalized model to optimal performance before starting the adversarial training. Compared to training D_2 at the beginning, it is easier to improve the performance of the adversarial training.

7.4 Experimental Results

In this section, we evaluate the performance of WL and AdvPL using three real-world datasets. In Section 7.4.1, we present experimental setup details including dataset descriptions, hyperparameters, and baselines. In Section 7.4.2, we present our main results of the accuracy comparison of our proposed framework and other baseline models and the corresponding training efficiency on three datasets. In Section 7.4.3, we present detailed analysis on the performance of the proposed framework using the first two datasets. First, we compare the performance of the personalized model (without requested similar data) and global model to demonstrate the benefits of the personalized model. Second, we measure the performance improvement of the personalized model, WL and AdvPL, with requested similar data. Third, we compare our algorithms with other similarity metrics and demonstrate the advantages of our proposed framework. Fourth, we conduct sensitivity analysis and investigate the effects of the probability distribution of the requested data and the budget size on the performance of WL and AdvPL.

7.4.1 Experimental Setup

Datasets: To evaluate the performance of our algorithm, we conduct our experiments on three real-world datasets, including UNIX Command Sequence, Shakespeare Text and Yesi-Well health data. The UNIX Command Sequence dataset [139] is composed of 50 files, where each file corresponds to one user's command sequence collected by the UNIX acct auditing mechanism. Each user is recorded with a long sequence consisting of the UNIX command in a period of time, such as *troff, dpost, eqn, sed, cat, ls, gs* and so forth. The sequence length of each user is 15000 and the average number of command types for these 50 users is over

100. We split the long command sequence of each user evenly into 500 sequences. For each sequence, we aim to build a classifier that predicts the final command based on the previous commands in this sequence. More specifically, the input data is a sequence consisting of UNIX commands with length T and the task is to predict the next command at the $T+1$ step. It is natural that different users have their own typing styles. For example, technical users and non-technical users often have different command sequences. In our experiment, we build one personalized model for each user. The Shakespeare Text dataset is constructed from The Complete Works of William Shakespeare [18]. This dataset is written in the form of plays and each speaking role in the plays is treated as an individual user. We subsample 40 speaking roles and build one personalized model for each speaking role. For each speaking role, we process its input text data to a sequence list where each sequence has a fixed length 100. The numbers of sequences of each speaking role are within the range between 5000 and 10000. The task is to predict the next character after reading previous characters in each sequence. YesiWell health dataset [140] was collected between 2010 and 2011 as a collaboration study by different institutes, including PeaceHealth Laboratories, SK Telecom Americas, and the University of Oregon, to help people maintain active lifestyles and lose weight. The dataset includes a group of 254 overweight and obese individuals and records the information of various aspects such as physical activities, social activities, biomarkers, and biometric measures. The total distance of walking and running is included in physical activities and measured via a mobile device carried by each user. The distance of each user is reported daily and forms a long sequence. Our task is to build a classifier for each user to predict the daily distance based on the distance sequence of previous days. For this classification task, we divide the distance into 30 ranges and assign a label to each distance range. After preprocessing, we select 69 users by removing sequences with missing values

and abnormal patterns due to data reporting error. The sequence length is set as 10 and the data size of each user is between 200 and 500. We summarize the characteristics of UNIX Command Sequence, Shakespeare Text and YesiWell data in Table 7.1.

Table 7.1: The characteristics of UNIX Command Sequence, Shakespeare Text and YesiWell data.

Dataset	**Users**	**Sequence Length**	**Data Size**
UNIX Command Sequence	50	50	500
Shakespeare Text	40	100	5000 - 10000
YesiWell data	69	10	200 - 500

Hyperparameters: In our experiment, we use two-layer long short-term memory (LSTM) neural networks as the classification model for both two tasks. Each layer of the LSTM classification model has the dimension 256 and the output of the second LSTM layer is sent to a softmax output layer for prediction. For each user, we select 80% of the sequences as training data and the rest as testing data. We also choose LSTM to build the auto-encoder which is composed of an encoder and a decoder. Both the encoder and a decoder consist of two-layer LSTMs. The dimensions of the hidden layer in the encoder and decoder are both set as 256. Each subsequence is embedded with 512 dimensions before sending to the encoder as input. The last hidden layer of the encoder is taken out to be the input of the decoder. For the GAN model, both the discriminator and generator are feedforward neural networks. More specifically, the generator has two hidden layers with dimensions 50 and 100, respectively. The discriminator also has two hidden layers with dimensions 100 and 50, respectively. The dimension of the Gaussian noise is the same as the size of the hidden representations by the encoder.

Baselines: In our proposed WL (AdvPL), each user collects $\mathcal{D}_s$ from other users and trains his personalized model using $\mathcal{D}_i$ and $\mathcal{D}_s$ in a weighted training (adversarial training) manner.

We compare our WL and AdvPL with the following baselines:

- Global: the global model is a one-fit-all model for all of the users, which is built on the collection of all users' training data.
- PL: each user trains his own personalized model (the same structure as the global model) only based on his own training data.
- PL_Rand: each user randomly requests $\mathcal{D}_s$ and trains the model simply with the combination of $\mathcal{D}_i$ and $\mathcal{D}_s$.
- PL_Euc: each user requests $\mathcal{D}_s$ from other users based on the euclidean similarity metric and trains his model as PL_Rand. The euclidean similarity metric is computed as follows: for each user, we compute the one-hot-vector of each command and obtain a vector v_i by averaging the one-hot-vector of all commands. Similarly, for each subsequence of other users, we compute its vector v_s. Then we can compute the euclidean similarity metric between v_i and v_s.
- PL_Cos: each user requests $\mathcal{D}_s$ from other users based on the cosine similarity metric and trains his model as PL_Rand. The computation of the cosine similarity metric is the same as PL_Euc.
- PL_Multi: each user requests $\mathcal{D}_s$ using our proposed autoencoder and GAN. We apply the multi-task framework [141] and treat $\mathcal{D}_i$ and $\mathcal{D}_s$ as two different tasks. In this implementation, only the final classification layer for $\mathcal{D}_i$ and that for $\mathcal{D}_s$ are different.

We run our methods and all baselines for five times and report the mean and standard deviation of accuracy in our evaluation.

7.4.2 Main Results

7.4.2.1 Accuracy Comparison

The accuracy comparison of our proposed framework and other baseline models on three datasets is shown in Table 7.2. The size of requested data $\mathcal{D}_s$ is 1000 for UNIX Command Sequence, 8000 for Shakespeare Text and 300 for YesiWell. From the experimental results, we summarize key observations and leave the detailed comparison in the following sections. First, the average accuracy of PL is higher than that of Global, indicating that one-fit-all model may miss the specific features of individual's data, especially in the scenarios where the distribution of individual's data is rather diverse. Second, our proposed framework outperforms other similarity metrics in selecting similar data. As we explained, the auto-encoder and GAN can capture implicit complex features from the data whereas other similarity metrics only compute simple statistical information. Third, the proposed AdvPL performs the best among all approaches. It demonstrates that AdvPL can reduce the distribution discrepancy between individual's data and requested data and thus improve the overall prediction accuracy.

We further test the statistical significance of the improvements between our proposed methods and baseline models. We run our methods and all baseline models for five times, use the independent two-sample t-test, and then calculate the p-value. For UNIX dataset, the p-values of testing AdvPL against Global, PL, PL_Rand, PL_Euc, PL_Cos, PL_Multi, and WL are 0.0006, 0.0073, 0.0045, 0.0102, 0.0143, 0.1819, and 0.1619, respectively. Using the threshold of 0.05, AdvPL has statistically significant improvement over Global, PL, PL_Rand, PL_Euc and PL_Cos. AdvPL can still achieve decent p-values (less than 0.2) when comparing with WL and PL_Multi (using GAN based similarity metric). For Shakespeare and YesiWell

Table 7.2: Accuracy comparison (mean ± std) of the proposed framework and baselines based on five runs on UNIX Command Sequence, Shakespeare, and YesiWell. The scale of the numbers is %.

Dataset	PL	Global	PL_Rand	PL_Euc	PL_Cos	PL_Multi	WL	AdvPL
UNIX	54.86 ± 3.42	51.98 ± 2.89	54.46 ± 3.16	55.88 ± 2.77	56.10 ± 3.05	59.46 ± 3.17	59.34 ± 2.96	**61.42 ± 3.26**
Shakespeare	53.67 ± 2.25	51.87 ± 2.37	52.60 ± 2.07	53.91 ± 2.12	53.88 ± 2.27	55.49 ± 1.96	56.19 ± 2.21	**57.45 ± 1.97**
YesiWell	32.79 ± 4.19	30.48 ± 4.37	32.19 ± 3.67	34.21 ± 3.94	34.62 ± 3.77	35.46 ± 4.01	35.82 ± 4.23	**37.84 ± 3.75**

Table 7.3: Training time (second) of the proposed framework and other baselines.

Dataset	PL	Global	PL_Rand	PL_Euc	PL_Cos	PL_Multi	WL	AdvPL
UNIX	118.2	1053.7	211.5	598.2	701.6	543.2	533.5	581.2
Shakespeare	677.7	4762.5	1213.2	4078.2	4983.4	2785.3	2752.4	3042.5
YesiWell	65.3	584.7	118.9	313.5	372.9	252.7	259.3	273.2

datasets, we have similar observations.

7.4.2.2 Training Efficiency

The training efficiency is an important metric to evaluate the performance of the proposed framework. In this section, we compare the training time of our proposed framework and baselines. We report the results in Table 7.3 and summarize the key findings as the following. First, we can see that the training of auto-encoder and GAN can increase the total completion time from the comparison between PL_Rand and WL. However, this extra training time is worth as we demonstrate that the proposed framework can improve the overall performance significantly. Second, other similarity metrics, PL_Euc and PL_Cos, need to compare paired data one by one, which incurs high computational cost. Although auto-encoder and GAN will increase the total training time in our framework, each data sample only needs one single comparison when using the discriminator of the trained GAN for similar data selection. Third, adversarial learning takes more time to finish the training than WL, but the burden increased by the adversarial learning is not significant.

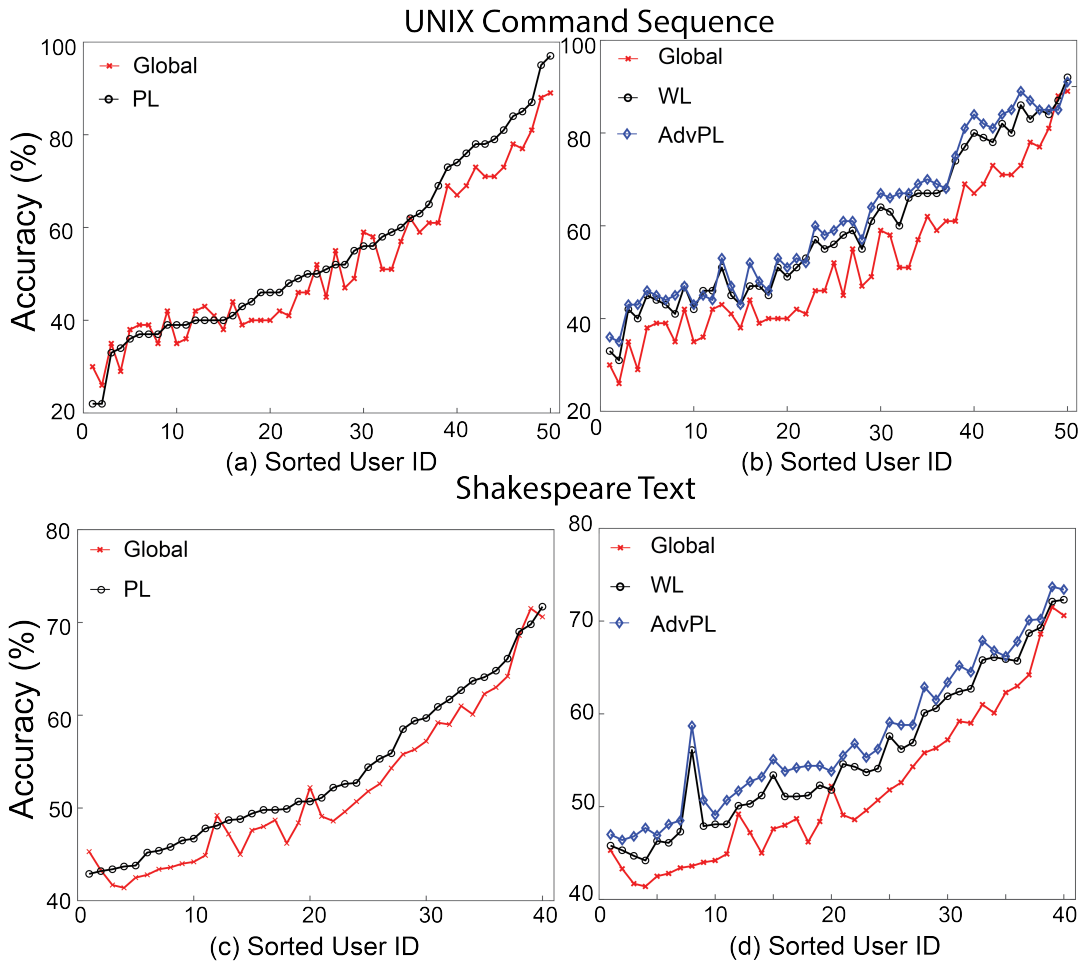

Figure 7.3: UNIX Command Sequence (a): The accuracy of the Global and PL for each user. (b): The accuracy of the Global, WL and PL_Adv for each user. Shakespeare text (c): The accuracy of the Global and PL for each user. (d): The accuracy of the Global, WL and PL_Adv for each user.

7.4.3 Detailed Performance Analysis

In this section, we conduct detailed evaluations and compare the performance of WL, AdvPL and other baseline models. We study the effects of probability distribution and requested dataset size on the performance of the proposed framework. For demonstration purpose and space limit, we only show the performance analysis on UNIX Command Sequence and Shakespeare Text.

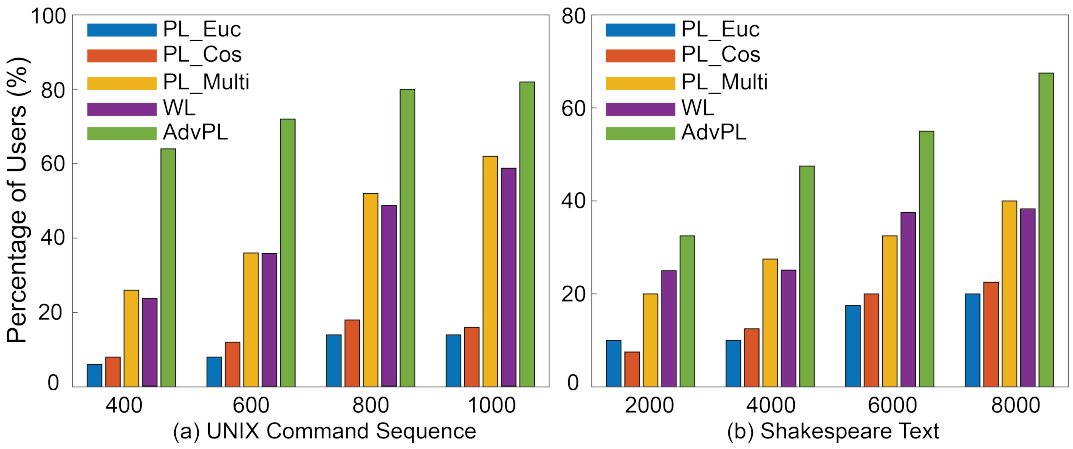

Figure 7.4: The percentage of users with accuracy increment over 5% using the PL_Euc, PL_Cos, PL_multi, WL and AdvPL compared to the PL.

7.4.3.1 Accuracy Comparison of Global and PL

In this experiment, we compare the performance of the Global and PL for each user to show the necessity of building the personalized model. The experimental result for UNIX Command Sequence (Shakespeare Text) is shown as Figure 7.3(a)(Figure 7.3(c)). To illustrate the result more clearly, we sort the users according to the prediction accuracy of the personalized model. The accuracy of each user using the Global (PL) is shown as the red star line (black dot line). It can be seen that the trend of the PL is above the Global for most users. More specifically, the average accuracy of the Global and PL for UNIX Command Sequence (Shakespeare Text) is 51.98% (51.87%) and 54.86% (53.67%), respectively. The maximum accuracy increment of the PL over the Global for UNIX Command Sequence (Shakespeare Text) is 8.0% (3.7%). As aforementioned, Global captures the overall information of all training samples and overlooks the user-specific information. However, the user-specific information is the key to improve the performance of the PL. Hence the PL can better learn the pattern of each user.

7.4.3.2 Accuracy Comparison of WL and AdvPL

In this experiment, we test the effectiveness of the proposed algorithms and show the accuracy of the WL and AdvPL for UNIX Command Sequence (Shakespeare Text) as Figure 7.3(b) (Figure 7.3(d)). The size of requested $\mathcal{D}_s$ for UNIX Command Sequence (Shakespeare Text) is set as 1000 (8000). The accuracy of the WL (AdvPL) is shown as the black dot line (blue triangular line). For comparison, we also plot the accuracy of the Global as the red star line.

We have the following observations. First, the accuracy of the AdvPL is higher than that of the WL. It demonstrates that adversarial learning can minimize the distribution discrepancy between $\mathcal{D}_i$ and $\mathcal{D}_s$. The reduced distribution discrepancy can help the AdvPL achieve higher accuracy than the WL. Second, both WL and AdvPL have a great advantage over the Global. It shows that our proposed algorithms are effective to select similar data for each user and help improve the performance of the personalized model. More specifically, the average accuracy of the WL and AdvPL for UNIX Command Sequence (Shakespeare Text) is 7.36% (4.32%) and 9.44% (5.58%) higher than that of the Global, respectively. In contrast, the average accuracy of the PL for UNIX Command Sequence (Shakespeare Text) is only 2.88% (1.80%) higher than that of the Global.

We further investigate the effects of the assigned weights in WL and compare its performance with PL_Sim (unweighted learning as shown in Equation 7.5). From our experimental results, the average performance of WL and PL_Sim over all users are at the same level. This is because for most of the users in our experiments, the probability scores of their requested data $\mathcal{D}_s$ are high, i.e., approaching to 1, and the loss function (Equation 7.7) of WL is reduced to the loss function (Equation 7.5) of PL_Sim. However, for a few other users,

Table 7.4: Accuracy comparison (mean ± std) based on five runs for UNIX Command Sequence using different similarity metrics and $\mathcal{D}_s$ sizes. The average accuracy of the Global and PL is: 51.98%, 54.86%, respectively. The scale of the numbers is %.

$\lvert\mathcal{D}_s\rvert$	PL_Rand	PL_Euc	PL_Cos	PL_Multi	WL	AdvPL
400	54.48 ± 3.23	55.12 ± 3.04	55.34 ± 2.91	56.84 ± 3.27	56.82 ± 3.15	**60.14 ± 3.77**
600	54.24 ± 3.19	55.28 ± 3.55	55.62 ± 2.89	57.96 ± 3.02	57.72 ± 3.37	**60.68 ± 2.75**
800	54.62 ± 3.43	55.56 ± 3.24	55.78 ± 2.10	58.62 ± 3.22	58.48 ± 2.88	**61.10 ± 3.73**
1000	54.46 ± 3.16	55.88 ± 2.77	56.10 ± 3.05	59.46 ± 3.17	59.34 ± 2.96	**61.42 ± 3.26**

Table 7.5: Accuracy comparison (mean ± std) based on five runs for Shakespeare Text using different similarity metrics and $\mathcal{D}_s$ sizes. The average accuracy of the Global and PL is: 51.87%, 53.67%, respectively. The scale of the numbers is %.

$\lvert\mathcal{D}_s\rvert$	PL_Rand	PL_Euc	PL_Cos	PL_Multi	WL	AdvPL
2000	52.36 ± 2.04	53.63 ± 2.23	53.45 ± 1.83	54.21 ± 2.19	54.32 ± 2.57	**56.62 ± 2.79**
4000	52.49 ± 2.57	53.65 ± 1.79	53.61 ± 2.18	54.70 ± 2.28	54.95 ± 2.62	**56.91 ± 2.42**
6000	52.35 ± 1.95	53.79 ± 2.17	53.80 ± 2.33	55.24 ± 2.29	55.64 ± 1.93	**57.31 ± 2.14**
8000	52.60 ± 2.07	53.91 ± 2.12	53.88 ± 2.27	55.49 ± 1.96	56.19 ± 2.21	**57.45 ± 1.97**

the probability scores of the requested data $\mathcal{D}_s$ are mixed, i.e., some data samples have lower probability scores while other data samples have higher probability scores. In this case, the performance of WL is better than Pl_Sim. Taking one user (ID = 3) from UNIX Command Sequence as an example, the accuracy of WL is 3% higher than that of PL_Sim. We provide more detailed analysis in 7.4.3.4 and discuss under what scenarios WL can outperform PL_Sim.

7.4.3.3 Accuracy Comparison Using Different Similarity Metrics

Previous section shows the performance between the WL and AdvPL. In this section, we compare with the personalized models using other similarity metrics. For better comparison, we show the average accuracy rather than plotting the accuracy of all users. The comparison result for UNIX Command Sequence (Shakespeare Text) is shown as the 5th row in Table 7.4 (Table 7.5) with $|\mathcal{D}_s| = 1000$ ($|\mathcal{D}_s| = 8000$). We have the following

interesting observations. First, if the user randomly requests $\mathcal{D}_s$, the average performance of the PL_Rand is slightly decreased compared to the PL. Second, the performance of the PL_Euc and PL_Cos helps improve the performance over the PL, however, the performance increment of our proposed WL is the best among these three similarity metrics. Third, the AdvPL achieves the best performance among all strategies. PL_Multi adopts the general multi-task learning framework [141] to learn common features between $\mathcal{D}_i$ and $\mathcal{D}_s$, however, it only achieves similar performance as WL, which demonstrates that our proposed AdvPL is more effective than the traditional multi-task learning method.

Moreover, we also study the effect of requested dataset $\mathcal{D}_s$ size on the performance of the WL and AdvPL. The result for UNIX Command Sequence (Shakespeare Text) is shown in Table 7.4 (Table 7.5) with $\mathcal{D}_s$ size increasing from 400 to 1000 (2000 to 8000). We have the following three observations from the results. First, the performance of the PL_Rand is slightly decreased compared to the PL under different $\mathcal{D}_s$ sizes. It is reasonable that large distribution discrepancy of the randomly requested $\mathcal{D}_s$ and $\mathcal{D}_i$ can deteriorate the personalized model performance. Second, the accuracy values of the PL_Euc, PL_Cos, PL_Multi and WL increase with the increasing size of $\mathcal{D}_s$. Third, we discover that with smaller $\mathcal{D}_s$ size, the accuracy increment from WL to AdvPL is more significant. More specifically, the accuracy increment of the AdvPL over the WL for UNIX Command Sequence (Shakespeare Text) is 3.32% (2.30%), 2.96% (1.96%), 2.62% (1.67%) and 2.08% (1.26%), respectively, with the corresponding $\mathcal{D}_s$ size as 400 (2000), 600 (4000), 800 (6000), 1000 (8000). The reason is that with smaller $\mathcal{D}_s$ size, some less similar data in $\mathcal{D}_s$ impedes the model performance improvement to a greater extent due to the smaller total amount of data. In contrast, if the total data size is larger, then the effects of some less similar data is smaller and the advantage of the AdvPL is also weakened. As a result, the AdvPL plays a more important

role for personalized model with smaller budget size of $\mathcal{D}_s$. It is more practical in real scenarios that one user can only request limited amount of data from other users due to the privacy concern or communication cost burden.

To further compare the performance of different strategies, we plot the percentage of users with accuracy increment greater than 5% using the PL_Euc, PL_Cos, PL_Multi, WL and AdvPL over the PL under different $\mathcal{D}_s$ size. The result is shown in Figure 7.4. It can be seen that the AdvPL greatly outperforms other strategies. In addition, the WL achieves better performance than the PL_Euc and PL_Cos, demonstrating the effectiveness of our proposed method in requesting similar data.

7.4.3.4 Effects of Probability Distribution and Budget Size on WL, AdvPL and PL_Sim

In this section, we investigate the effects of the probability distribution of the requested data determined by D_2 and the budget size of $\mathcal{D}_s$ for the WL, AdvPL and PL_Sim. We randomly select a single user (ID = 29) in the UNIX Command Sequence dataset and conduct the experiments for demonstration purpose. Table 7.6 shows the effects of probability distribution of the requested data for the WL, AdvPL and PL_Sim. We divide the range of probability score by D_2 evenly into five regions between 0.5 and 1.0. For each region, we select $\mathcal{D}_s$ containing 1000 samples and test the accuracy of the WL, AdvPL and PL_Sim. For example, if we select the range [0.5, 0.6], it means the probability score of all requested data falls into the range [0.5, 0.6].

We have the following observations. First, the accuracy of the WL is lower if the probability distribution of the data in $\mathcal{D}_s$ is in a low range. For example, if all of the data in $\mathcal{D}_s$ is requested within the range [0.9, 1.0], then the accuracy of the WL is 5% higher than

that of the $\mathcal{D}_s$ within the range [0.5, 0.6]. As we know, higher probability by the discriminator indicates the data is more likely to be sampled from $\mathcal{D}_i$. Consequently, lower probability distribution range of $\mathcal{D}_s$ will cause larger distribution discrepancy with $\mathcal{D}_s$ and lower performance improvement of the WL. Second, the accuracy of the AdvPL for $\mathcal{D}_s$ with different probability distribution ranges is stable. The advantage of the AdvPL is that it can reduce the distribution discrepancy between $\mathcal{D}_i$ and $\mathcal{D}_s$, so that the performance of the AdvPL can still be improved greatly under a higher distribution discrepancy. It demonstrates that the AdvPL can better improve the personalized model performance if only less similar data is available. Third, the advantage of the AdvPL is weakened if the distribution discrepancy between $\mathcal{D}_i$ and $\mathcal{D}_s$ is smaller. The reason is that if $\mathcal{D}_s$ is of high similarity compared with $\mathcal{D}_i$, then the user can directly combine them and train the model without considering the effect of distribution discrepancy.

We also investigate the difference between WL and PL_Sim. In fact, both WL and PL_Sim are sensitive on the probability score. When the probability score is within low range, the accuracy of WL is higher than that of PL_Sim. This is because that WL lowers the importance of the requested data while PL_Sim treats all requested data with uniform weight. Taking the probability score within the range [0.5, 0.6] as an example, the accuracy values (mean $\pm$ std) of PL_Sim and WL are 60.8 $\pm$ 1.64 and 63.0 $\pm$ 1.94, respectively. Its p-value from the t-test is 0.0230, which indicates a significant improvement of WL over PL_Sim. When the probability score is within the high range, WL and PL_Sim have similar performance because WL is reduced to PL_Sim as aforementioned. In short, WL outperforms PL_Sim when the similarity between requested data and original data is relatively low. WL and PL_Sim achieve similar performance when sufficient similar data can be collected from

Table 7.6: Accuracy comparison (mean ± std) based on five runs for a single user (UNIX Command Sequence) based on $\mathcal{D}_s$ with different discriminator score.

Discriminator Score	PL_Sim	WL	AdvPL
0.5 - 0.6	60.8 ± 1.64	63.0 ± 1.94	66.8 ± 1.79
0.6 - 0.7	63.0 ± 2.05	63.2 ± 1.48	67.0 ± 2.12
0.7 - 0.8	64.6 ± 1.67	66.0 ± 1.41	68.2 ± 2.17
0.8 - 0.9	67.0 ± 1.49	67.2 ± 1.10	67.8 ± 2.16
0.9 - 1.0	66.8 ± 1.31	67.0 ± 1.58	68.2 ± 1.92

Table 7.7: Accuracy comparison (mean ± std) based on five runs for a single user (UNIX Command Sequence) based on $\mathcal{D}_s$ with different size.

$\mathcal{D}_s$ Size	PL_Sim	WL	AdvPL
400	57.6 ± 1.51	59.2 ± 1.48	63.0 ± 2.00
600	58.8 ± 1.64	60.4 ± 1.81	64.0 ± 2.44
800	60.2 ± 1.48	60.2 ± 1.09	65.4 ± 1.87
1000	60.8 ± 1.64	63.0 ± 1.94	66.8 ± 1.79

neighbors.

Table 7.7 shows the effects of $\mathcal{D}_s$ size on the performance of the PL_Sim, WL and AdvPL. To test the sensitivity of the adversarial training process, we request less similar data and set the probability distribution of the data in $\mathcal{D}_s$ within the range [0.5, 0.6]. We have the following observations. First, the accuracy of the WL increases slowly with the increasing size of $\mathcal{D}_s$ as we select less similar data with the probability distribution in the low range [0.5, 0.6]. So although the user requests more data, the performance gain by these increasing less similar data is still insignificant. Second, with the adversarial training, the accuracy of the AdvPL improves greatly over the WL. It can be seen that the performance improvement of the AdvPL is almost stable under different sizes of $\mathcal{D}_s$. It demonstrates that the AdvPL can still improve the model performance greatly even with a limited budget size

and less similar data available.

7.5 Summary

In this paper, we proposed the AdvPL framework enabling individual users to effectively collect data from other users and train a robust personalized model. We proposed to use the auto-encoder and GAN to select similar data from other users. The trained auto-encoder and GAN, which capture inherent information of user's personal data, can be efficiently used to choose similar data from others, thereby avoiding tedious process of paired data comparison. We have developed two approaches to combine the requested data and user's own data to improve the performance of personalized learning. The first approach is weighted learning that assigns different weights to different requested data. Then the model can capture the importance of different requested data. The second approach is adversarial training that maps selected data and user's own data to the same feature space and jointly trains the personalized model. The adversarial training can effectively mitigate potential distribution discrepancy between selected data and user's own data. We conducted extensive experiments to demonstrate the effectiveness of the proposed framework.

8 Conclusion and Future Work

8.1 Conclusion

In this dissertation, we focus on the fairness-aware machine learning under distribution shift. We have done several works to address the following problems:

- How to achieve fairness and maintain good prediction performance in federated learning setting, where the distribution of the test data is unknown;
- How to achieve fairness and prediction performance for classification under sample selection bias;
- How to achieve fairness for regression problem under sample selection bias;
- How to request similar data for an individual user in a distributed setting and how to use the requested data to build a personalized model.

To address the above problems, we summarize the work we have done as the following.

In Chapter 4, we have proposed a fairness-aware agnostic federated learning framework to deal with unknown testing data distributions. We applied kernel reweighing functions to parametrize the loss function and fairness constraint. Hence our framework can achieve both good model accuracy and fairness on unknown testing data. We conducted a series of experiments on two datasets and experimental results demonstrated three benefits of the trained centralized model by our fairness-aware agnostic federated learning. First, it can improve the prediction accuracy under the distribution shift from the training data to the testing data. Second, it can guarantee fairness on the unknown testing data. Third, it can guarantee the fairness of each local client.

In Chapter 5, we have proposed a robust and fair learning framework to deal with the sample selection bias. Our framework adopts the reweighing estimation approach for bias correction and the minimax robust estimation for achieving robustness on prediction accuracy and fairness on test data. We further developed two algorithms to handle sample selection bias when test data is both available and unavailable. Experimental results showed our algorithms can achieve both good prediction accuracy and fairness on test data.

In Chapter 6, we have developed a framework for fair regression under sample selection bias when dependent variable values of a set of samples are missing. The framework adopts the classic Heckman model to correct sample selection bias and captures a variety of fairness notions via inequality and equality constraints. We applied the Lagrange duality theory to derive the dual convex optimization and showed the conditions of achieving strong duality for fairness metrics in our framework. For the two popular fairness notions, mean difference and mean squared error difference, we further derived explicit formulas without optimizing iteratively. Experimental results on three real-world datasets demonstrated our approach's effectiveness.

In Chapter 7, we proposed the AdvPL framework enabling individual users to effectively collect data from other users and train a robust personalized model. We proposed to use the auto-encoder and GAN to select similar data from other users. The trained auto-encoder and GAN, which capture inherent information of user's personal data, can be efficiently used to choose similar data from others, thereby avoiding tedious process of paired data comparison. We have developed two approaches to combine the requested data and user's own data to improve the performance of personalized learning. The first approach is weighted learning that assigns different weights to different requested data. Then the model can capture the importance of different requested data. The second approach is adversarial training that

maps selected data and user's own data to the same feature space and jointly trains the personalized model. The adversarial training can effectively mitigate potential distribution discrepancy between selected data and user's own data. We conducted extensive experiments to demonstrate the effectiveness of the proposed framework.

8.2 Future Work

In this section, we point out some future directions for fairness-aware machine learning under distribution shift.

Following the research in Chapter 4, we can extend our agnostic fair framework to cover other commonly used fairness notations. e.g., equalized odds and equalized opportunity [13], and incorporate surrogate functions in agnostic fair constraints of our framework to address the challenge of the indicator function used in fairness notations. We will also study kernel function parametrization with different basis functions. Our proposed framework can also be adapted to the centralized fairness-aware learning where the training and testing data differ. Moreover, the proposed framework can also be applied in the fair transfer learning where distribution shift usually exists between the source domain and target domain.

Following the research in Chapter 5, we can study other types of sample selection bias, i.e., missing not at random that the sample selection probability also depends on the label. We will also study how to enforce other fairness notions such as equal opportunity [14].

Following the research in Chapter 6, we will conduct theoretical analysis and empirical evaluation of density based fairness notions, e.g., SP and BGL, and notions for multiple sensitive attributes. Some recent work [142] proposed to use Hirschfeld-Gebelein-Rényi Maximum (HGR) correlation coefficient as a regression fairness notion to evaluate the independence be-

tween prediction and sensitive attributes. However, it is quite challenging to compute HGR. We can only get analytical solution for some certain distributions, e.g., jointly Gaussian distribution [143], or apply approximation approaches. In our future work, we will study HGR in our framework. We will also study improved estimators [144] that address the limitations of Heckman estimator, e.g., sensitivity of estimated coefficients with respect to the distributional assumptions on the error terms, and extend to nonlinear cases, e.g., kernel regression, in our fair regression.

Following the research in Chapter 7, we have two major directions for our future work. First, we will study how to achieve privacy in our AdvPL. The auto-encoder and GAN contain private information of the individual user's data. Previous works demonstrate that deep learning models can memorize abundant information of the training data [145]. To protect the privacy of training data in $\mathcal{D}_i$, we can build differential privacy preserving versions of auto-encoder and GAN, e.g., by adopting the ideas of [146] and [147] respectively. Moreover, the data $\mathcal{D}_s$ collected from other users are also private and users may not want to share. We will study the use of local differential privacy [148] for private data comparison and collection. Second, we will investigate how to determine most appropriate data to improve the performance of personalized model. Our current work is based on the idea of selecting similar data to boost the performance. Ideally, we want to determine the properties of new data that can best improve the model performance, e.g., reducing the prediction error of the built model. With that, we only need to collect truly useful data and skip redundant ones, which will greatly reduce the communication burden and improve efficiency. One idea is to use active learning to select new data that improve personalized model performance. However, existing active learning strategies [149] may not be directly applied here because the data determined by the active learning may contain large distribution discrepancy from

individual user's own data. It is interesting to investigate how to combine the active learning and personalized model.

Bibliography

[1] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[2] E. Cambria and B. White, "Jumping nlp curves: A review of natural language processing research," *IEEE Computational intelligence magazine*, vol. 9, no. 2, pp. 48–57, 2014.

[3] Y. Goldberg, "Neural network methods for natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–309, 2017.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016.

[5] P. Druzhkov and V. Kustikova, "A survey of deep learning methods and software tools for image classification and object detection," *Pattern Recognition and Image Analysis*, vol. 26, no. 1, pp. 9–15, 2016.

[6] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[7] J. Larson, S. Mattu, L. Kirchner, and J. Angwin, "Compas dataset," https://github.com/propublica/compas-analysis, 2017.

[8] J. Quiñonero-Candela, M. Sugiyama, N. D. Lawrence, and A. Schwaighofer, *Dataset shift in machine learning*. Mit Press, 2009.

[9] C. Cortes, M. Mohri, M. Riley, and A. Rostamizadeh, "Sample selection bias correction theory," in *ALT*, 2008.

[10] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," 2018.

[11] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3752–3761.

[12] T. Tommasi, M. Lanzi, P. Russo, and B. Caputo, "Learning the roots of visual domain shift," in *European Conference on Computer Vision*. Springer, 2016, pp. 475–482.

[13] M. Hardt, E. Price, N. Srebro *et al.*, "Equality of opportunity in supervised learning," in *NeurIPS*, 2016.

[14] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[15] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," in *AISTATS*, 2015.

[16] L. Zhang, Y. Wu, and X. Wu, "Achieving non-discrimination in prediction," in *IJCAI*, 2017.

[17] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 259–268.

[18] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2016.

[19] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *NIPS*, 2017.

[20] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM TIST*, 2019.

[21] J. J. Heckman, "Sample selection bias as a specification error," *Econometrica: Journal of the econometric society*, pp. 153–161, 1979.

[22] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[23] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *NIPS*, 2014.

[24] S. Hajian and J. Domingo-Ferrer, "A methodology for direct and indirect discrimination prevention in data mining," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 7, pp. 1445–1459, 2012.

[25] Y. Wu and X. Wu, "Using loglinear model for discrimination discovery and prevention," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2016, pp. 110–119.

[26] F. Kamiran and T. Calders, "Data preprocessing techniques for classification without discrimination," *Knowledge and Information Systems*, vol. 33, no. 1, pp. 1–33, 2012.

[27] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *ACM KDD*, 2015.

[28] I. Žliobaite, F. Kamiran, and T. Calders, "Handling conditional discrimination," in *IEEE ICDM*, 2011.

[29] F. P. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney, "Optimized pre-processing for discrimination prevention," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 3995–4004.

[30] D. Xu, S. Yuan, L. Zhang, and X. Wu, "Fairgan: Fairness-aware generative adversarial networks," in *IEEE Bigdata*, 2018.

[31] H. Zhao and G. Gordon, "Inherent tradeoffs in learning fair representations," in *NeurIPS*, 2019.

[32] J. Song, P. Kalluri, A. Grover, S. Zhao, and S. Ermon, "Learning controllable fair representations," in *AISTATS*, 2019.

[33] M. B. Zafar, I. Valera, M. Gomez Rodriguez, and K. P. Gummadi, "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment," in *ACM WWW*, 2017.

[34] M. Donini, L. Oneto, S. Ben-David, J. S. Shawe-Taylor, and M. Pontil, "Empirical risk minimization under fairness constraints," in *NIPS*, 2018.

[35] A. Cotter, H. Jiang, S. Wang, T. Narayan, S. You, K. Sridharan, and M. R. Gupta, "Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals," *Journal of Machine Learning Research*, 2019.

[36] T. B. Hashimoto, M. Srivastava, H. Namkoong, and P. Liang, "Fairness without demographics in repeated loss minimization," in *ICML*, 2018.

[37] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro, "Learning non-discriminatory predictors," in *COLT*, 2017.

[38] S. Baharlouei, M. Nouiehed, and M. Razaviyayn, "R\'enyi fair inference," in *ICLR*, 2020.

[39] Y. Wu, L. Zhang, and X. Wu, "On convexity and bounds of fairness-aware classification," in *ACM WWW*, 2019.

[40] M. P. Kim, A. Ghorbani, and J. Zou, "Multiaccuracy: Black-box post-processing for fairness in classification," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019, pp. 247–254.

[41] M. Sugiyama, S. Nakajima, H. Kashima, P. Von Buenau, and M. Kawanabe, "Direct importance estimation with model selection and its application to covariate shift adaptation." in *NIPS*, vol. 7. Citeseer, 2007, pp. 1433–1440.

[42] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," in *ICML*, 2004.

[43] A. J. Storkey and M. Sugiyama, "Mixture regression for covariate shift," *Advances in neural information processing systems*, vol. 19, p. 1337, 2007.

[44] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola, "Correcting sample selection bias by unlabeled data," *Advances in neural information processing systems*, vol. 19, pp. 601–608, 2006.

[45] Y. Yu and C. Szepesvári, "Analysis of kernel mean matching under covariate shift," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. [Online]. Available: http://icml.cc/2012/papers/330.pdf

[46] A. Liu and B. Ziebart, "Robust classification under sample selection bias," in *NeurIPS*, 2014.

[47] J. Wen, C.-N. Yu, and R. Greiner, "Robust learning under uncertain test distributions: Relating covariate shift to model misspecification." in *ICML*, 2014.

[48] W. Hu, G. Niu, I. Sato, and M. Sugiyama, "Does distributionally robust supervised learning give robust classifiers?" in *International Conference on Machine Learning*. PMLR, 2018, pp. 2029–2037.

[49] X. Chen, M. Monfort, A. Liu, and B. D. Ziebart, "Robust covariate shift regression," in *Artificial Intelligence and Statistics*. PMLR, 2016, pp. 1270–1279.

[50] Y. Wang, A. Kucukelbir, and D. M. Blei, "Robust probabilistic modeling with bayesian data reweighting," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3646–3655.

[51] B. Taskesen, V. A. Nguyen, D. Kuhn, and J. Blanchet, "A distributionally robust approach to fair classification," *arXiv preprint arXiv:2007.09530*, 2020.

[52] A. Rezaei, R. Fathony, O. Memarrast, and B. D. Ziebart, "Fairness for robust log loss classification," in *AAAI*, 2020.

[53] M. Yurochkin, A. Bower, and Y. Sun, "Training individually fair ML models with sensitive subspace robustness," in *ICLR*, 2020.

[54] C. Schumann, X. Wang, A. Beutel, J. Chen, H. Qian, and E. H. Chi, "Transfer of machine learning fairness across domains," *arXiv preprint arXiv:1906.09688*, 2019.

[55] A. Coston, K. N. Ramamurthy, D. Wei, K. R. Varshney, S. Speakman, Z. Mustahsan, and S. Chakraborty, "Fair transfer learning with missing protected attributes," in *AIES*, 2019.

[56] P. Lahoti, A. Beutel, J. Chen, K. Lee, F. Prost, N. Thain, X. Wang, and E. H. Chi, "Fairness without demographics through adversarially reweighted learning," in *NeurIPS*, 2020.

[57] N. Kallus and A. Zhou, "Residual unfairness in fair machine learning from prejudiced data," in *ICML*, 2018.

[58] A. Blum and K. Stangl, "Recovering from biased data: Can fairness constraints improve accuracy?" in *FORC*, 2020.

[59] H. Jiang and O. Nachum, "Identifying and correcting label bias in machine learning," in *AISTATS*, 2020.

[60] D. Madras, E. Creager, T. Pitassi, and R. Zemel, "Learning adversarially fair and transferable representations," in *ICML*, 2018.

[61] W. Du, D. Xu, X. Wu, and H. Tong, "Fairness-aware agnostic federated learning," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 181–189.

[62] H. Singh, R. Singh, V. Mhasawade, and R. Chunara, "Fairness violations and mitigation under covariate shift," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 3–13.

[63] A. Rezaei, A. Liu, O. Memarrast, and B. Ziebart, "Robust fairness under covariate shift," in *AAAI*, 2021.

[64] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *NeurIPS*, 2012.

[65] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *OSDI*, 2014.

[66] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *NeurIPS*, 2017.

[67] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "Adacomp: Adaptive residual gradient compression for data-parallel distributed training," in *AAAI*, 2018.

[68] S. Wang, A. Pi, X. Zhao, and X. Zhou, "Scalable distributed dl training: Batching communication and computation," in *AAAI*, 2019.

[69] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *NeurIPS*, 2018.

[70] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM CCS*, 2017.

[71] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *ICLR*, 2017.

[72] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *NIPS*, 2017.

[73] A. T. Suresh, F. X. Yu, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *ICML*, 2017.

[74] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 3365–3375.

[75] S. Zheng, Z. Huang, and J. Kwok, "Communication-efficient distributed blockwise momentum sgd with error-feedback," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 446–11 456.

[76] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, 2019.

[77] N. Agarwal, A. T. Suresh, F. Yu, S. Kumar, and B. McMahan, "cpsgd: Communication-efficient and differentially-private distributed sgd," in *NIPS*, 2018.

[78] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *ACM CCS*, 2015.

[79] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 1–11.

[80] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," in *ICLR*, 2016.

[81] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2512–2520.

[82] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *ICLR*, 2020.

[83] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[84] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[85] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *ICML*, 2019.

[86] A. Beutel, J. Chen, Z. Zhao, and E. H. Chi, "Data decisions and theoretical implications when adversarially learning fair representations," in *FAT/ML*, 2017.

[87] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[88] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[89] P. D. Grünwald, A. P. Dawid *et al.*, "Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory," *The Annals of Statistics*, 2004.

[90] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[91] W. Du and X. Wu, "Fair and robust classification under sample selection bias," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2999–3003.

[92] J. Blanchet, Y. Kang, and K. Murthy, "Robust wasserstein profile inference and applications to machine learning," *Journal of Applied Probability*, 2019.

[93] P. Laforgue and S. Clémençon, "Statistical learning from biased training samples," *arXiv preprint arXiv:1906.12304*, 2019.

[94] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, 1979.

[95] T. Calders, A. Karim, F. Kamiran, W. Ali, and X. Zhang, "Controlling attribute effect in linear regression," in *2013 IEEE 13th international conference on data mining*. IEEE, 2013, pp. 71–80.

[96] K. D. Johnson, D. P. Foster, and R. A. Stine, "Impartial predictive modeling: Ensuring group fairness in arbitrary models," *arXiv e-prints*, pp. arXiv–1608, 2016.

[97] J. Komiyama, A. Takeda, J. Honda, and H. Shimao, "Nonconvex optimization for regression with fairness constraints," in *International conference on machine learning*. PMLR, 2018, pp. 2737–2746.

[98] A. Agarwal, M. Dudík, and Z. S. Wu, "Fair regression: Quantitative definitions and reduction-based algorithms," in *Proceedings of the 36th International Conference on Machine Learning ICML*, vol. 97. PMLR, 2019, pp. 120–129.

[99] D. Steinberg, A. Reid, S. O'Callaghan, F. Lattimore, L. McCalman, and T. S. Caetano, "Fast fair regression via efficient approximations of mutual information," *CoRR*, vol. abs/2002.06200, 2020.

[100] W. Du, X. Wu, and H. Tong, "Fair regression under sample selection bias," *arXiv preprint arXiv:2110.04372*, 2021.

[101] J. Fitzsimons, A. Al Ali, M. Osborne, and S. Roberts, "A general framework for fair regression," *Entropy*, vol. 21, no. 8, p. 741, 2019.

[102] J. Mary, C. Calauzenes, and N. El Karoui, "Fairness-aware learning for continuous attributes and treatments," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4382–4391.

[103] E. Chzhen, C. Denis, M. Hebiri, L. Oneto, and M. Pontil, "Fair regression with wasserstein barycenters," *arXiv preprint arXiv:2006.07286*, 2020.

[104] W. Du and X. Wu, "Fair and robust classification under sample selection bias," in *Proceedings of the 2021 ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2021.

[105] H. Narasimhan, A. Cotter, M. Gupta, and S. Wang, "Pairwise fairness for ranking and regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[106] R. Berk, H. Heidari, S. Jabbari, M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth, "A convex framework for fair regression," in *FAT ML*, 2018.

[107] C. Zhao and F. Chen, "Unfairness discovery and prevention for few-shot regression," in *2020 IEEE International Conference on Knowledge Graph (ICKG)*. IEEE, 2020, pp. 137–144.

[108] T. Le Gouic and J.-M. Loubes, "Computing the price for fairness in a regression framework," *arXiv preprint arXiv:2005.11720*, 2020.

[109] E. Chzhen, C. Denis, M. Hebiri, L. Oneto, and M. Pontil, "Fair regression via plug-in estimator and recalibration with statistical guarantees," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[110] D. Alabi, N. Immorlica, and A. Kalai, "Unleashing linear optimizers for group-fair learning and optimization," in *Conference On Learning Theory, COLT 2018, Stockholm, Sweden, 6-9 July 2018*, ser. Proceedings of Machine Learning Research, S. Bubeck, V. Perchet, and P. Rigollet, Eds., vol. 75. PMLR, 2018, pp. 2043–2066. [Online]. Available: http://proceedings.mlr.press/v75/alabi18a.html

[111] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach, "A reductions approach to fair classification," 2018.

[112] J. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores," *arXiv preprint arXiv:1609.05807*, 2016.

[113] http://archive.ics.uci.edu/ml/datasets/communities+and+crime, 2009.

[114] L. F. Wightman, "Lsac national longitudinal bar passage study. lsac research report series." 1998.

[115] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *AAAI*, 2015.

[116] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," in *AAAI*, 2017.

[117] D. H. Park, H. K. Kim, I. Y. Choi, and J. K. Kim, "A literature review and classification of recommender systems research," *Expert systems with applications*, 2012.

[118] Y. Cheng, F. Wang, P. Zhang, and J. Hu, "Risk prediction with electronic health records: A deep learning approach," in *SDM*, 2016.

[119] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

[120] W. Du and X. Wu, "Advpl: Adversarial personalized learning," in *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2020, pp. 90–98.

[121] ——, "Enhancing personalized modeling via weighted and adversarial learning," *International Journal of Data Science and Analytics*, vol. 12, no. 1, pp. 1–14, 2021.

[122] C. Che, C. Xiao, J. Liang, B. Jin, J. Zho, and F. Wang, "An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease," in *SDM*, 2017.

[123] Q. Suo, F. Ma, Y. Yuan, M. Huai, W. Zhong, A. Zhang, and J. Gao, "Personalized disease prediction using a cnn-based similarity learning method," in *IEEE BIBM*, 2017.

[124] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, and J. Sun, "Multi-layer representation learning for medical concepts," in *ACM KDD*, 2016.

[125] M. Huai, C. Miao, Q. Suo, Y. Li, J. Gao, and A. Zhang, "Uncorrelated patient similarity learning," in *SDM*, 2018.

[126] F. Wang, J. Sun, and S. Ebadollahi, "Composite distance metric integration by leveraging multiple experts' inputs and its application in patient similarity assessment," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2012.

[127] M. Li and L. Wang, "A survey on personalized news recommendation technology," *IEEE Access*, 2019.

[128] F. Luo, G. Ranzi, X. Wang, and Z. Y. Dong, "Social information filtering-based electricity retail plan recommender system for smart grid end users," *IEEE Transactions on Smart Grid*, 2017.

[129] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, and L. Getoor, "Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems," in *ACM RecSys*, 2015.

[130] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu, "Diversifying personalized recommendation with user-session context." in *IJCAI*, 2017, pp. 1858–1864.

[131] Z. Yu, J. Lian, A. Mahmoody, G. Liu, and X. Xie, "Adaptive user modeling with long and short-term preferences for personalized recommendation." in *IJCAI*, 2019, pp. 4213–4219.

[132] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE TPAMI*, 2013.

[133] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *IEEE CVPR*, 2015.

[134] A. H. Liu, Y.-C. Liu, Y.-Y. Yeh, and Y.-C. F. Wang, "A unified feature disentangler for multi-domain image translation and manipulation," in *NeurIPS*, 2018.

[135] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," in *ICLR*, 2017.

[136] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *IEEE CVPR*, 2016.

[137] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-level variational autoencoder: Learning disentangled representations from grouped observations," in *AAAI*, 2018.

[138] S. Narayanaswamy, T. B. Paige, J.-W. Van de Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, and P. Torr, "Learning disentangled representations with semi-supervised deep generative models," in *NeurIPS*, 2017.

[139] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theusan, Y. Vardi *et al.*, "Computer intrusion: Detecting masquerades," *Statistical science*, 2001.

[140] N. Phan, J. Ebrahimi, D. Kil, B. Piniewski, and D. Dou, "Topic-aware physical activity propagation in a health social network," *IEEE intelligent systems*, 2015.

[141] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[142] V. Grari, B. Ruf, S. Lamprier, and M. Detyniecki, "Fairness-aware neural rényi minimization for continuous features," in *IJCAI*, 2020.

[143] S. Asoodeh, F. Alajaji, and T. Linder, "On maximal correlation, mutual information and data privacy," in *2015 IEEE 14th Canadian workshop on information theory (CWIT)*. IEEE, 2015, pp. 27–31.

[144] P. Puhani, "The heckman correction for sample selection and its critique," *Journal of economic surveys*, vol. 14, no. 1, pp. 53–68, 2000.

[145] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *ACM CCS*, 2017.

[146] N. Phan, Y. Wang, X. Wu, and D. Dou, "Differential privacy preservation for deep auto-encoders: an application of human behavior prediction," in *AAAI*, 2016.

[147] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," *CoRR*, 2018.

[148] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *IEEE FOCS*, 2013.

[149] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.