MULTIOBJECTIVE OPTIMIZATION APPROACHES
FOR BIAS MITIGATION IN MACHINE LEARNING

A Dissertation in
Informatics
by
Mohammad Mahdi Kamani

© 2020 Mohammad Mahdi Kamani

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2020
The dissertation of Mohammad Mahdi Kamani was reviewed and approved by the following:

James Z. Wang  
Professor of Information Sciences and Technology  
Dissertation Co-Adviser  
Co-Chair of Committee

Mehrdad Mahdavi  
Assistant Professor of Computer Science and Engineering  
Dissertation Co-Adviser  
Co-Chair of Committee

Lee C. Giles  
David Reese Professor at the College of Information Sciences and Technology

Xiang Zhang  
Associate Professor of Information Sciences and Technology

Chris Forest  
Professor of Meteorology and Atmospheric Science

Jia Li  
Professor of Statistics

Mary Beth Rosson  
Professor of Information Sciences and Technology  
Head of the Graduate Programs at the College of Information Sciences and Technology
Achieving astounding progress and evolution during the past few decades, Artificial Intelligence (AI) is becoming omnipresent in every aspect of our lives. Nowadays, AI systems are touching human lives on so many unimaginable levels, through which they are transforming our prior way of life. Despite these tremendous achievements, some challenges hinder the effectiveness of these systems in real-world applications. Bias in learning systems is one of the greatest challenges these AI algorithms are facing, which has induced both technical and social complications. Bias could appear in a variety of types and from different sources, such as class-imbalance problem, fairness issues, noise in data or labels, spurious correlations between features and labels to name but a few. Regardless of the source or type of bias, it can degrade the quality of the learned model or become a source of discrimination against minority groups in sensitive features such as race, gender, or education. In some cases, the consequences of bias could be detrimental and affect numerous human lives. Hence, it is of paramount importance to account for these effects during the training procedure of machine learning models. There are various solutions for each of the biases individually, but there is not a framework that can be adapted to a number of these biases together. In addition, there might be a trade-off between satisfying the main goal of the learning and addressing the bias issues at the same time. Thus, we ought to find solutions that entail the optimal trade-off or compromise between these conflicting objectives, which is not always considered by prior solutions.

In an attempt to mitigate the effects of bias in learning tasks, in this dissertation, two multiobjective optimization approaches are proposed. The overarching goal is to reduce different bias problems into these two frameworks and solve them using proposed gradient-based algorithms. The first approach dubbed as Targeted Data-driven Regularization employs a small well-crafted target dataset that is free of bias and uses it in a bilevel programming problem to prevent the main learning process to drift toward a biased model. The inner level in this bilevel programming is designed to learn parameters of the model, and the outer level aims at finding the optimal weights for different groups or classes in the dataset using that target dataset. An application of this approach is
presented for the class-imbalance problem with both synthetic and real-world empirical results that outperform the state-of-the-art.

The second approach called *Pareto descent optimization* designed to solve multiobjective optimizations involving both the main learning objective and the bias mitigation objectives. The gradient-based algorithm is guaranteed to converge to Pareto stationary points of the problem with optimal compromises. A novel extension of this algorithm is proposed to trace points from the Pareto frontier of the problem and can converge to a solution with desired levels of compromises between objectives. This approach is generic and can be generalized to any other multiobjective optimization. This idea is utilized for the fairness problem in both supervised and unsupervised learning settings. Empirical results for both these applications assert that they can superbly exceed the state-of-the-art performance in this domain.
# TABLE OF CONTENTS

List of Figures ix

List of Tables xv

Acknowledgments xvii

Chapter 1
Introduction 1
  1.1 Motivation ................................. 1
  1.2 Bias ........................................ 2
  1.3 Multiobjective Approaches ...................... 5
  1.4 Contributions ................................ 7
  1.5 Outline ...................................... 8
    1.5.1 Notation .................................. 10

Chapter 2
Targeted Data-driven Regularization 11
  2.1 Introduction .................................. 11
  2.2 Related Work .................................. 14
  2.3 Targeted Data-driven Regularization .............. 15
    2.3.1 The proposed framework ..................... 15
    2.3.2 Stochastic bilevel optimizer ................. 17
    2.3.3 Weights Interpretation ........................ 20
  2.4 Time Complexity Comparison ...................... 21

Chapter 3
Application I: Class-Imbalance Problem 24
  3.1 Experimental Results .......................... 24
  3.2 Synthetic data .................................. 25
    3.2.1 MNIST Dataset ............................. 25
3.2.2 CIFAR10 Dataset ........................................... 28
3.3 Severe weather prediction dataset ................................ 30

Chapter 4
Pareto Descent Optimization ................................. 33
4.1 Pareto Efficiency .............................................. 35
4.2 Learning Pareto Efficient Classifiers ......................... 37
4.2.1 Bilevel Optimization ...................................... 38
4.3 Pareto Frontier .................................................. 42
4.3.1 Pareto Frontier Geometry ................................ 42
4.3.2 Pareto Frontier Extraction ................................ 44
4.3.2.1 Preference-Based Pareto Descent Optimization .... 46
4.4 Convergence Analysis .......................................... 54
4.5 Final Remarks .................................................... 58
4.6 Missing Proofs ................................................. 59
4.6.1 Proof of Theorem 1 ......................................... 59
4.6.1.1 Proof of Theorem 1 .................................. 60
4.6.2 Proof of Lemma 3 ........................................... 62
4.6.3 Proof of Lemma 4 ........................................... 63
4.6.4 Proof of Theorem 3 ......................................... 65
4.6.5 Proof of Theorem 4 ......................................... 67

Chapter 5
Application II: Pareto Efficient Fairness in Supervised Learning 69
5.1 Introduction ...................................................... 69
5.2 Additional Related Work ..................................... 73
5.3 Pareto Efficient Fairness ....................................... 76
5.3.1 Reduction of known notions of fairness to PEF .......... 77
5.3.1.1 Pareto efficient equality of opportunity ................. 77
5.3.1.2 Pareto efficient equalized odds ......................... 78
5.3.1.3 Pareto efficient disparate mistreatment ............... 78
5.4 Experimental Results ......................................... 79
5.5 Final Remarks .................................................... 84

Chapter 6
Application III: Pareto Efficient Fairness in Dimension Reduction 87
6.1 Introduction ...................................................... 87
6.2 Related Work .................................................... 90
6.3 Problem Formulation .......................................... 92
6.3.1 PCA .......................................................... 92
6.3.2 Fair PCA ..................................................... 93
6.4 Pareto Fair Subspace .......................................... 95
6.4.1 Relaxation methods and their limitations ............... 95
6.4.2 Pareto fair subspace ....................................... 97
| 6.4.3 | Proof of existence | 98 |
| 6.5 | Adaptive Optimization | 99 |
| 6.5.1 | Gradient descent for PCA | 99 |
| 6.5.2 | Pareto fair PCA | 100 |
| 6.5.3 | Comparison with other approaches | 103 |
| 6.6 | Experiment | 104 |
| 6.6.1 | Binary sensitive feature | 105 |
| 6.6.2 | Multiple group sensitive feature | 107 |
| 6.6.3 | Fairness in composition | 108 |
| 6.7 | Final Remarks | 110 |
| 6.8 | Missing Proofs | 112 |
| 6.8.1 | Proof of Theorem 5 | 113 |
| 6.8.2 | Proof of Lemma 5 | 114 |
| 6.8.3 | Proof of Theorem 6 | 116 |
| 6.8.4 | Proof of Theorem 7 | 119 |

Chapter 7

Concluding Remarks and Future Work 121

Appendix A

Bow Echo Dataset 124

A.1 Introduction 124

A.2 Bow Echo Feature Extraction 126

A.2.1 Radar Images and Regions of Interest 127

A.2.2 Skeletonization 128

A.2.3 Skeleton Pruning using Fuzzy Logic 131

A.2.3.1 Main Skeleton Fuzzy Inference System 133

A.2.3.2 Branch Fuzzy Inference System 135

A.2.3.3 Importance Value 136

A.2.3.4 Pruning Algorithm 139

A.2.4 Skeleton Context 140

A.3 Bow Echo Classification and Detection 143

A.3.1 Shape Matching with Skeleton Context 143

A.3.2 Neighboring Effect 145

A.3.3 Distance Definition and Prototype Extraction 146

A.3.4 Classifier 148

A.4 Case Study 150

A.4.1 Training Data 150

A.4.2 Prototype Extraction 150

A.4.3 Classification 151

Appendix B

Datasets Meta-data 152
2.1 In standard (unweighted) binary classification with unbalanced data between two classes (Class 1 versus Class 2), the decision boundary is mostly determined based on the majority class’s descent direction. In TDR, a small and unbiased target dataset is used, with which each class’s gradients based on their inner products with the gradient of the target dataset can be reweighted. See Section 2.3.3 for more details. 

2.2 The proposed targeted data-driven regularization for out-of-distribution generalization.

3.1 Test accuracy on imbalanced dataset containing two classes (‘4’ and ‘9’) of MNIST. The test dataset is a balanced dataset of the two classes. Total number of training samples is 5,000 and class 4 is in the smaller class with the imbalance ratio of $\rho = 0.005$. Target distribution has 50 balanced samples. Targeted meta learning outperforms example reweighting, both in terms of accuracy and speed.

3.2 Recall rate of the smaller class 4 during the training of the experiment in Figure 3.1 with imbalance ratio $\rho = 0.005$. Similar to test accuracy, TDR outruns example reweight in recall rate.

3.3 Comparing the test accuracy of TDR with example reweighting for different imbalance ratios from $\rho \in \{0.1, 0.05, 0.02, 0.005\}$. The dataset is MNIST with two classes (4 and 9), with the same setting as in Figure 3.1. Each experiment is repeated 5 times to examine the robustness of algorithms.

3.4 (a) The test accuracy of TDR vs. example reweighting for different imbalance ratios $\rho \in \{0.1, 0.05, 0.02, 0.005\}$ with long-tailed MNIST dataset. Each experiment is repeated 5 times. (b) Distribution of learned weights for each class for $\rho = 0.02$. Each row shows the distribution of weights for each class during training. The size of each class is decreased exponentially, however, the learned weights are not exactly proportional to the inverse of their size.
3.5 Comparing the test accuracy of TDR with example reweighting, hard weighting with inverse of frequency, random weights and no reweighting. These experiments are run for different imbalance ratios from $\rho \in \{0.3, 0.2, 0.1, 0.05\}$. The dataset is CIFAR10 with two classes (4 and 9) and the model is a 4-layer ConvNet. Each experiment is repeated 5 times to examine the robustness of algorithms. The detailed results are in Table 3.1.

3.6 A NEXRAD level III radar image of the United States with a bow echo on May 24, 2008, 10:20 GMT. The bow echo part is being magnified.

3.7 Accuracy, recall, and precision rates on balanced test dataset after 11 epochs of training for targeted-meta learning, hard reweighting with inverse of frequency, and without weights. The training dataset contains the complete radar images from year 2008 with $\rho = 0.017$, while the test set is a balanced dataset of bow echo and non bow echo samples from year 2009. Test accuracy and recall rate on bow echo samples reach to 0.8605 and 0.855 respectively. With hard weights the recall rate increases but at the cost of decreasing precision.

4.1 Hypothetical trade-off between accuracy and any fairness constraint. $w_1^*$ is the optimum solution that minimizes the empirical risk, while $w_2^*$ is the optimum solution to satisfy the fairness constraint. Each solution on the Pareto frontier cannot be dominated in terms of both empirical risk and the fairness constraint by any other feasible solution.

4.2 Schematic view of relationship between weighting parameters and Pareto optimal solution properties. If we replace $\alpha$ in a bi-objective optimization problem with $\theta = \frac{\sin \theta}{\sin \theta + \cos \theta}$, it can be shown that for every $\theta \in [0, \frac{\pi}{2}]$ the Pareto efficient solution $w_\theta$ is the tangent point of the Pareto frontier curve and the line perpendicular to the rotated coordinated of $h_1(w)$ with angle $\theta$, i.e. $h_\theta(w) = \cos \theta \cdot h_1(w) + \sin \theta \cdot h_2(w)$.

4.3 Example objectives in the parameter space. The red line is the projection of Pareto frontier on the parameter space.

4.4 The relation between an arbitrary Pareto frontier with two different preference lines parameterized by their $\pi$ or $p$ vectors. The Pareto frontier is from the bi-objective problem defined in (4.19) with $w \in \mathbb{R}^2$. The desired solution for each line is on the intersection of that preference line and the Pareto frontier. For these two lines these solutions are $w_{PB_1}$ and $w_{PB_2}$ in the figure. It can be inferred that when we increase one objective’s preference value, the desired solution is getting closer to that objective’s minimizer.
4.5 Using objectives in Example 1 with $w \in \mathbb{R}^{20}$ to (a) demonstrate two challenging cases, which cannot converge to the desired point $w^*_{PB}$ by only using $h_{PB}$ as the objective; and (b) how $PB-PDO$ by adaptively choosing objectives can achieve this goal. In (a) we converge to points $w_I$ and $w_{II}$ for cases I and II, respectively, which are stationary points of the $h_{PB}$ but they are not the desired point. .......................... 51

4.6 Comparing the convergence trajectory of our Algorithm 2 (PDO), the preference-based algorithm introduced in [108], and our proposed Algorithm 3 (PB-PDO), when they all start from the same initial solution $w_0$. PDO uses descent directions based on $h$ to reach to a point from the Pareto frontier. The preference-based approach introduced in [108] uses ascent and descent directions to converge to the desired point and not other points from the Pareto frontier. On the other hand, PB-PDO using descent directions on $h_{PB}$ to converge to the desired points, while tracing other points from the Pareto frontier during the convergence. .......................... 54

4.7 Value of $h_1(w)$ versus $h_2(w)$ for different $w$s for arbitrary parameters in Example 2. .......................... 59

5.2 The results of applying Algorithm 2 to satisfy the Pareto efficient equality of opportunity for (a) Adult dataset with gender as its sensitive feature, (b) COMPAS dataset with race as its sensitive feature (c) COMPAS dataset with sex as its sensitive feature. The drop in accuracy is small in all cases, while the goal of equal opportunity is satisfied perfectly. By looking at false positive rates, it seems that this notion is capable of satisfying equalized odds as well as equality of opportunity. .......................... 81

5.3 Adult dataset with race as its sensitive feature, which has 5 groups. In this case, our proposed algorithm can superbly satisfy the property of Pareto efficient equality of opportunity, despite the fact that we have 11 objectives in the minimization. The groups are AIE: Amer-Indian-Eskimo, API: Asian-Pac-Islander, B: Black, O: Other, W: White, and T represents the total. .......................... 82

6.1 Applying normal PCA and fair PCA to the Adult dataset with gender as its sensitive feature. The first two figures show the reconstruction error of normal PCA (trained on all data) applied to each group, fair PCA (trained on all data) applied to each group, and normal PCA trained on the data of each group individually. The last figure reveals the difference between normal and fair PCA reconstruction loss on all data, which is very tiny and negligible. .......................... 106
6.2 Comparing Pareto fair PCA algorithm introduced in this chapter (pairwise and single disparity error) with fair PCA algorithms using SDP relaxation introduced in [134] and [113]. The experiment is on binary features of Adult and Credit datasets (gender and sex). The average disparity error of algorithms on the Adult dataset clearly shows the superiority of the Pareto fair PCA with pairwise disparity error objectives and then the single disparity error objectives. In the Credit dataset, Pareto fair PCA with pairwise objectives has a slightly better performance with respect to two other methods.

6.3 Comparing the total reconstruction loss of Pareto fair PCA algorithm introduced in this chapter (with pairwise objectives), with fair PCA algorithms using SDP relaxation introduced in [134] and [113]. The experiment is on binary features of Adult and Credit datasets (gender and sex). From both figures, it can be inferred that Pareto fair PCA satisfies the fairness objective better, while it incurs a tiny increase in the total reconstruction loss with respect to the normal PCA. On the other hand, Fair PCA with SDP suffers from a huge increase in the total reconstruction loss to satisfy the fairness objective.

6.4 Applying normal and fair PCA on the Adult dataset with “race” as its sensitive feature. Each plot shows the reconstruction error of the normal PCA (trained on the whole data) applied to each group’s data, fair PCA (trained on the whole data) applied to each group’s data, and normal PCA trained on each group’s data individually.

6.5 Disparity error of normal and fair PCA trained on the Adult dataset with “race” as its sensitive feature. Each plot depicts the disparity errors of different groups with normal and fair PCA.

6.6 Applying normal and fair PCA on the Adult dataset with “race” as its multiple group sensitive feature. Left shows the difference between reconstruction loss of fair and normal PCA, which is infinitesimal. On the other hand, the right plot shows their difference in terms of average disparity error, which is huge and demonstrating the efficacy of fair PCA in addressing fairness even in multiple group sensitive feature cases.

6.7 Applying normal and fair PCA on the Credit dataset with “Marriage” as its sensitive feature. Each plot shows the reconstruction error of the normal PCA (trained on the whole data) applied to each group’s data, fair PCA (trained on the whole data) applied to each group’s data, and normal PCA trained on each group’s data individually. The last figure shows the reconstruction error of normal PCA and fair PCA on this dataset with multiple group sensitive feature of marriage.

6.8 Disparity error of normal and fair PCA trained on the Credit dataset with “Marriage” as its sensitive feature. Each plot depicts the disparity errors of different groups with normal and fair PCA. The last figure shows the average of disparity errors across groups.
6.9 The effect of Pareto fair PCA on a downstream SVM classification task. The first row is using the Adult dataset with gender as its sensitive feature, and the second row is using the Credit dataset with sex as its sensitive feature. In both datasets, we reduce their feature space dimension to 10 once using normal PCA and once using Pareto fair PCA. Then use the new representation to learn a linear SVM. Column (a) is the accuracy among different groups, (b) is the true positive rate, and (c) is the DEO introduced by [38] (the lower is better), all on the test dataset. It clearly can be noted that applying Pareto fair PCA can reduce the gap between true positive rates of different groups and enhance the fairness of downstream models.

A.1 The complete scheme for automatic bow echo detection. In the first stage, it extracts regions of interest, which are red parts in radar images. Then in the skeletonization step, it extracts skeleton map of each part, and prunes it using the proposed algorithm. After finding pruned skeleton, it computes skeleton context and finds the nearest match in the database of bow echo prototypes. Finally, it uses mixture discriminant analysis classifier to detect whether it is a bow echo or not.

A.2 Radar image of the United States Continent with a bow echo, May 2, 2008, 07:10 GMT. We magnify the part that bow echo happened (i.e. red regions).

A.3 Different stages in the skeletonization process. (a) We start from extracting the binary image of the object. (b) We apply Euclidean Distance Transform on the binary image. (c) Using (A.2) calculate the flux map. (d) Apply a threshold to extract the initial skeleton. (e) and (f) are the output of Algorithm 5, which extracts vertices and edges of the initial skeleton.

A.4 Inputs and the output membership functions of Main Skeleton fuzzy inference system for a sample skeleton.

A.5 Inputs and the output membership functions of Branch fuzzy inference system for a sample skeleton.

A.6 A simple skeleton graph with 13 vertices and 11 edges. Colored vertices are included in the calculation of the importance value of $V_1$ when $\Omega = 2$.

A.7 Belief Window (BW) versus $\Delta$, and $B_T$. The slope of change in BW size would decrease by increasing $\Delta$. This would be intensified by decreasing branch tolerance ($B_T$).

A.8 Output of pruning algorithm: (a) main skeleton degree of belief, (b) branch degree of belief, (c) Pruned skeleton. In (a) and (b) images are in grayscale, and higher values represent higher degree of beliefs.

A.9 Skeleton Context of 2 points on different skeletons that are matched based on the algorithm.

A.10 Bow echo prototypes with $k = 8$. Each column shows a bow echo prototype with its extracted skeleton.
A.11 Accuracy Rate versus (a) number of bow echo prototypes, and (b) number of mixture distributions in each class.
3.1 Experiment results on imbalanced CIFAR10 dataset with two classes of “4” and “9” (“4” is the minority class). Two different models are used, namely, ResNet20 and a simple 4-layer ConvNet with 2 fully connected layers as the classifier. TDR is compared with example reweighting, hard weights with inverse of class frequency and no weights or standard training.

3.2 Experimental results on long-tailed CIFAR10, with imbalance ratio from $\rho \in \{0.1, 0.2\}$ with two models, ResNet20 and ConvNet. We can use $n_i = n_0 \mu^i$ to decrease the class size and make the dataset long-tail. TDR achieves the best accuracies in ResNet20, however the training without any weights is doing better in ConvNet, which is not a suitable model for training of this dataset.

5.1 Comparison of the proposed PDO algorithm with FERM [38] and reductions [2]. Experiments are repeated multiple times with reporting the average and variance. Blue colors show the best results for DEO measure, which in all cases belong to the proposed Pareto descent algorithm. Moreover, in terms of accuracy, the Pareto descent algorithm outperforms both state-of-the-art algorithms.

6.1 Comparison of time complexity of different fair PCA algorithms to achieve an $\epsilon$-fair subspace (please see Section 6.3 for definition). Here $d$ denote the dimension of the original data, $r$ is the target dimension, and $k$ is the number of sensitive groups. Note that unlike previous studies that necessitates learning a subspace with larger dimension to guarantee fairness, our solution learns an exact $r$ dimensional subspace by imposing additional constrains captured by a new notion of fairness proposed in this work to distinguish between local optimal fair subspaces (we used the following abbreviations above, SDP: Semi-Definite Programming, MW: Multiplicative Weight Algorithm, LP: Linear Programming, GD: Gradient Descent). Note that all the algorithms have an initial step of finding the optimal rank $r$ subspace for each group, in which its time complexity is not included here.
B.1 Adult train dataset with gender as sensitive feature . . . . . . . . . . . . 152
B.2 Adult test dataset with gender as sensitive feature . . . . . . . . . . . . . 152
B.3 Adult train dataset with race sensitive feature. AIE: Amer-Indian-Eskimo,
   API: Asian-Pac-Islander . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
B.4 Adult test dataset with race sensitive feature. AIE: Amer-Indian-Eskimo,
   API: Asian-Pac-Islander . . . . . . . . . . . . . . . . . . . . . . . . . . . 153
B.5 COMPAS dataset with sex as sensitive feature . . . . . . . . . . . . . . . 153
B.6 COMPASS dataset with race as sensitive feature . . . . . . . . . . . . . . . 153
First and foremost, my deepest appreciation belongs to my beautiful wife and loving friend, Artmiz. During the past eight years and all the ups and downs we have faced, she was a wonderful partner in my life, who not only supported me through her many sacrifices but also made me a better version of myself in every imaginable aspect of my life. My gratitude and respect for her are beyond words.

I would also like to thank my parents, Javad and Nahid, who always put my interests and well-being before theirs. During the past five years, I was not able to meet them in person due to the cruel travel ban, but I felt their presence every step of the way. This feeling gave me the hope and courage for a better future. I am thankful for their support and patience, and I am deeply sorry for the pain they have suffered during these years.

I thank my advisor, Prof. James Wang, for his continuous support and guidance he provided during my Ph.D. studies. His motivation, enthusiasm, and immense knowledge were my true encouragement in my research. Also, I want to offer my sincere gratitude to my co-advisor, Prof. Mehrdad Mahdavi, for his belief in me and the self-confidence he instilled in me. Through his boundless patience, guidance, and mentorship, I learned a lot. Without his support, this dissertation was not possible. I will always be indebted to him in all my current and future research endeavors.

I would like to thank my dissertation committee members for their helpful comments and suggestions on the dissertation: Prof. Lee Giles, Prof Xiang Zhang, Prof. Chris Forest, and Prof. Jia Li.

Last but by no means the least, I am grateful for my research colleagues and friends, without whom this was not possible. Especially, Farzin Haddadpour, Sadegh Farhang, Farshid Farhat, and Yuyang Deng, with whom I worked closely and enjoyed every moment of it.

This dissertation is based upon work supported by the National Science Foundation under Grant No. 1027854 and also used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation under grant number ACI-1548562. The findings and conclusions in this dissertation do not necessarily reflect the view of the funding agency.
CHAPTER 1

INTRODUCTION

1.1 Motivation

Tremendous advancements in the computational power of modern processors over the past few decades have made Artificial Intelligence (AI) ubiquitous in every walk of our lives. The field of AI has attested enormous progress in various directions, and it is growing at an unprecedented pace. These algorithms are vastly used in many systems and transforming human lives in countless ways. From smartphones and self-driving cars to smart home appliances, and from healthcare, finance, and transportation, to criminal justice and national security, AI is playing a critical role in decision-making processes, which demonstrates the extent to which human lives are dependent on these systems, and this is monotonously increasing every day.

However, there is no such thing as *free lunch*, meaning, these great advancements in AI come at a price. The societal impacts and consequences of these algorithms can be devastating and overwhelming that in some cases they can outcast their benefits altogether. *Algorithmic bias, privacy concerns, transparency of models, data access problem, AI ethics, and fairness* are just a handful of issues raising concerns about the use of these algorithms in different scenarios. For instance, recently IBM, Microsoft, and Amazon (during the recent Black Lives Matter movement) announced that they will not let law enforcement units to use their facial recognition systems, due to concerns about the misuse of these applications\(^1\). Researchers believe that these algorithms are biased toward females and people of color and prone to a higher error rate for these groups.

\(^1\)IBM, Microsoft And Amazon Not Letting Police Use Their Facial Recognition Technology.
compared to others, which could have destructive impacts, especially during this time. Hence, they suggest that these algorithms should be banned completely until they have effectively addressed the bias issue.

One of the main challenges that AI systems, and in particular machine learning models, are suffering from is the bias issue. Bias in any form, including but not limited to class-imbalance, noise in data or labels, fairness issues, and spurious correlations between features and labels, could potentially degrade the quality of the learned model for a specific task. In most cases, the problem stems from the bias data that later on used on training a machine learning model, and turns into a biased model. In some vital tasks such as pedestrian detection for autonomous vehicles, a failure due to biases in the model could be fatal [148]. This issue is exacerbated when the output of this biased model is used in decision-making systems, where they generate new data for training new models. For instance, it has been investigated that an AI system for assessing the risk of recidivism of defendants in courtrooms is 77% more likely to assign a higher risk of committing violent crimes in the future for black people than any other races [5]. Based on this assessment black defendants are more likely to be jailed for the same crime than other races, which unfairly creates biased data for new learning systems that are using the outcome of such decisions.

Hence, an important research question for this overwhelming concern in the field of AI is: How can we learn unbiased models from biased data?

To answer this essential question, researchers have provided different solutions for each type of bias individually. Each of these solutions is designed to address one of these biases and mostly cannot be generalized to other types. Thus, we need to understand how biases are affecting the training of a model and drifting it toward a biased model, and how different solutions are addressing it.

1.2 Bias

To better understand the similarities and differences between different types of biases, we explore some of these solutions for the four mentioned biases briefly.

Class-imbalance: In the class-imbalance problem, the main issue is that the majority of samples in a dataset belong to one or a number of classes, while the rest of the classes are represented with few samples each. In this case, the training is mostly dominated by those classes in the majority and the minority classes are getting ignored. In the end,
the learned model has superb performance over the majority classes’ samples, but it poorly performs on the learning task over the minority classes’ samples. However, since the number of samples in those minority classes are low, the incurred error compared to the average performance is negligible. An essential example of a learning task dealing with such a problem is critical incident detection and prediction. Whether it is detecting a natural phenomenon such as a severe weather incident, or it is a failure detection in a man-made system such as object detection in autonomous vehicles, those critical incidents happen rarely. As a result, the available dataset for such a task to learn from is heavily imbalanced toward normal incidents’ classes. Unfortunately, those critical incidents are the main reason behind using a learning algorithm in the first place, and the poor performance of the learned model over those samples can have detrimental consequences. Therefore, it is of paramount importance to be able to build an unbiased model that has an acceptable performance in detecting or predicting those incidents.

There are two categories of solutions for the class-imbalance problem, that are, pre-processing the data and cost-sensitive optimization. Pre-processing approaches include down-sampling the majority class and up-sampling or interpolating the minority class to generate new data points artificially. Generating new data points for minority classes can be done using new algorithms and ideas in generative adversarial neural networks [114, 99]. However, these solutions mostly suffer from a high level of noise, due to generating new data samples, or losing some information, by discarding some samples from the majority classes. Another strand of research for addressing this problem includes cost-sensitive approaches, where the goal is to penalize the cost of errors in those minority classes more than the majority classes’ [106, 160, 141]. If this idea pans out, it can address the problem of class-imbalance better than the other set of approaches. However, the difficulty of setting the best weight for each class or sample in the training is the greatest impediment to the generalization of this solution to different applications.

**Fairness:** Recently, the concerns regarding the fairness of AI algorithms in treating different sensitive groups is growing [41, 64, 5]. The problem is that models learned on biased data unfairly treat minority groups in sensitive features such as race, gender, or education level. Researchers argue that the outcome (or the preferential outcome) of these models should be independent of those features. For instance, in a university’s admission system it is reasonable that two persons with similar resumes should have equal chances of getting accepted, regardless of their race or gender. However, it has shown that these systems’ output, if trained oblivious to fairness criteria, mostly depends
on these sensitive features, which makes them biased. The problem aggravates when we understand that even by omitting those features the learned modes are still able to reveal them using some proxies through other features in the dataset, which makes this idea ineffective [11, 12, 59].

A recent study by Blum and Stangl [18] has shown that there are two sources of bias for the fairness issue, that are, under-representation and labeling bias. Under-representation is similar to class-imbalance problem, but for the sensitive groups instead of classes, where we have a smaller number of samples for some minority groups in the data that the model is going to be trained on. Labeling bias is due to corrupted data for any reason such as human cognitive bias in data collection. Hence, training a model on such data will transfer those biases to the resulting model.

Efforts on building a fairness-aware mechanism for decision-making systems require a proper notion of fairness to quantify fairness by taking into account the proper legal, ethical, and social context, and an algorithmic solution to alleviate the effects of biased data. For the first pillar, there have been numerous different notions and measures that quantify the degree of fairness in different models [64, 155, 154]. As for the second pillar, researchers have developed various algorithms such as constrained optimization to include the fairness criteria [38, 154], or minmax optimization to control the worst group’s fairness measure [2, 134] to name but a few. Most of these algorithmic solutions are designed for a limited setting, such as binary sensitive group, binary classification, or a certain class of loss functions, that is either impossible or nontrivial to adapt to other settings. Moreover, in most cases, these algorithmic solutions are computationally burdensome, which hinders their development to real practical settings. Finally, and more importantly, in most cases, they are not able to find points with optimal compromises between accuracy and fairness measures. Let alone finding points with the desired level of compromises between these two objectives.

Noise: The problem of noise is highly common and widespread not only in AI but also in any other field dealing with any form of data such as image, video, audio, text, to name but a few. In AI systems, the noise is added to the ground truth data either in the data collection process or in the labeling process (for the supervised learning tasks). Similar to the fairness problem, the noise could be due to the human cognitive bias either in the collection step or in the labeling step. It could also be due to the noise in sensory systems for data collection. In any case, the noise will corrupt the data or labels or both, which leads to a biased model at the end. Since there are various sources and types for
noise in datasets, there is quite a number of algorithmic solutions to address each of them. A remarkable strand of solutions is based on curriculum learning [17, 74], where the idea depends on the assumption that noisy data incurs a higher loss compared to the normal samples [63]. Based on this, the ideas are to either reweight those samples or present them later in the training pipeline to prevent the drift of the model toward a biased one. It is worth mentioning that this assumption is somehow in contrast with the one in the class-imbalance problem. In that case, those rare samples that incur a higher loss are the important ones and it is desired to up-weight their loss in the training, in order to maintain a good performance of the model over those samples. In the case of noise in data, on the other hand, it is desired to down-weight the noisy samples’ loss to prevent them from affecting the trained model.

**Spurious Correlation:** Another form of bias in AI systems is spurious correlations between some features in the training data and the labels, by which a model could be trained to wrongfully use them as a proxy for the label. A simple example is classifying cows versus camels using training images [14]. In this example, it has been shown that since most of the cow images are contain green backgrounds and camel images contain brown backgrounds, the classifier mostly relies on the background color to distinguish between these two classes. Hence, it could potentially fail to classify a cow in a sandy background correctly. To address this issue, Arjovsky et al. [7] and Ahuja et al. [3] propose a training procedure that guarantees the resulting model to be invariant to those features. The main idea is to train different models for each different environment of those features and aggregate the models at the end, by which under some conditions they ascertain the mentioned invariant characteristic.

### 1.3 Multiobjective Approaches

Having reviewed several different forms of biases, it seems that despite their differences in nature and their solutions, the key common ingredient in all of them is the necessity of satisfying multiple objectives at the same time. The main objective in most supervised machine learning algorithms is to minimize the empirical risk of the loss over training samples or maximizing the accuracy. For each of these biases, in addition to the main learning objective, it is required to satisfy a specific objective dealing with those biases in the training. For instance, in the class-imbalance problem, this extra objective could be the recall rate or true positive rate of those minority classes. In the fairness problem,
depending on the definition of fairness, the additional objective is changing to satisfy the requirement of that specific notion. For the noise problem, this objective could optimize the weights of different samples to abate the effects of noise in the training. Finally, for the spurious correlation problem, this objective is the empirical risk over each different environment.

Being multi-criteria in nature, we advocate multiobjective optimization approaches for mitigating biases in learning tasks. To that end, we propose two general multiobjective frameworks that can be applied to several different forms of biases, despite their differences in their goals. These two approaches are briefly discussed here:

- **Targeted Data-driven Regularization** [80, 82]: The idea behind this proposal is to use a small well-crafted target dataset that is free of those biases itself and can be utilized as a data-driven regularization for the main learning objective. In that sense, we propose a bilevel optimization framework, where its two levels of optimization are interleaved together. The inner level comprises the classical weighted cost function, where the weights are set for each group or class in the dataset. In this level, the parameters of the model are getting updated based on the learning objective. In the outer level, using the target dataset and its designated objective, the weights for each group or class are being updated. The nested loops of this optimization are designed to find the optimal weights for each group or class and parameters of the model simultaneously. To alleviate the high computational complexity of these nested loops, an stochastic approximation of this bilevel optimization is used.

- **Pareto Descent Optimization** [81]: Another idea to formulate the bias problem in learning tasks is to cast this problem as a vector optimization and directly solve the multiobjective optimization. The overarching goal of this optimization is to seek points with optimal trade-offs between the main learning objective and bias objectives. The points with optimal compromises between objectives are called Pareto efficient points of the problem, and a set of these points is called Pareto frontier. A novel bilevel and gradient-based algorithm is proposed in this part that converges to points from the Pareto frontier with optimal trade-offs. The main idea is to find a direction at every step that is descent with respect to every objective in the optimization. This will continue until such direction cannot be found, which is an indicator of a Pareto stationary point. In addition, this algorithmic solution has been extended to a novel gradient-based approach to trace different points on the
Pareto frontier. This approach is generic and can be utilized for any multiobjective optimization problem. Having access to this set and using a preference measure over different objectives, we can choose the best solution with the desired level of compromises.

1.4 Contributions

The main contributions of this dissertation can be summarized as follows:

- The proposed Targeted Data-driven Regularization approach can formulate the bias problem in a bilevel programming and efficiently address it by utilizing a small well-crafted target dataset that is free of those biases.

- An efficient stochastic approximation of the targeted data-driven regularization approach is proposed that can notably improve the computational complexity of the introduced bilevel programming.

- Extensive empirical results demonstrate the efficacy of the proposed targeted data-driven regularization approach on the class-imbalance problem. In this empirical study, both synthetic imbalanced datasets and a real-world severe weather dataset, which is heavily imbalanced, are employed.

- The proposed Pareto Descent Optimization formulates the bias problem as a multiobjective optimization and seeks Pareto efficient points of this vector optimization. A novel bilevel programming is proposed to solve this problem efficiently, which is a great tool for the convergence analysis of such algorithms. This gradient-based approach can be applied to any well-defined loss function that is trainable via gradient descent. The approach finds a direction that is a descent direction to all objectives at every iteration until it reaches a Pareto stationary point.

- A novel preference-based Pareto descent optimization is introduced that can effectively trace the points on the Pareto frontier of any multiobjective optimization problem. This framework not only find points with a desired level of compromises on the Pareto frontier but also traverse parts of the Pareto frontier while converging to that point. This characteristic enables us to track points from different parts of the Pareto frontier.
Pareto efficient fairness in supervised learning is proposed using the ideas from Pareto descent optimization. This novel proposal, despite some of the current state-of-the-art methods in this domain, can be applied to any well-defined notion of fairness in any setting (binary and non-binary sensitive feature, binary and non-binary classification, and with any loss function that is trainable by gradient descent). An extensive empirical study, using the Pareto descent optimization and its extension for finding points on the Pareto frontier, shows the efficacy of this approach that superbly improves the state-of-the-art results in this field.

Pareto efficient fairness in an unsupervised setting like dimension reduction is introduced. First, the problem is formulated as a strand of multiobjective optimization, and then, using the same approach, a gradient-based algorithm is proposed to find points from the Pareto frontier of the problem. An empirical study shows that this approach can greatly outperform the state-of-the-art and achieve a better trade-off between reconstruction loss and fairness measures. Moreover, from a computational perspective, the proposed algorithms are shown to be more efficient than state-of-the-art approaches. Finally, it has been empirically shown that enforcing fairness on the dimension reduction tasks using the proposed algorithm can improve the fairness criteria of the downstream classification tasks.

1.5 Outline

The outline of this dissertation is summarized as follows:

Chapter 2 In this chapter, the proposed targeted data-driven regularization framework is introduced and discussed. The details about the bilevel programming in this framework are discussed in Section 2.3. The stochastic approximation of this bilevel programming is explained in Section 2.3.2. In the inner level of this bilevel optimization, the parameters of the model are getting updated with a weighted loss function, with different weights for each group or class in the dataset. In the outer level, those weights are updated using the target dataset. The interpretation of the weights’ updates and how they are related to the gradients of each group as well as the target dataset is also presented in Section 2.3.3.

Chapter 3 In this chapter, an application of the targeted data-driven regularization approach is demonstrated practically on class-imbalance problems. To show the efficacy of this approach, different synthetic and real-world datasets are used. As for the real-world
problems, a severe weather detection task is being investigated, where the objective is to
detect bow shape patterns in radar images. This dataset is heavily imbalanced toward
samples without that specific bow pattern. Despite this fact, the targeted data-driven
regularization approach can learn a robust model for detecting these patterns with a
high recall rate.

Chapter 4  The proposed Pareto Descent Optimization approach using a bilevel pro-
gramming is presented and discussed in this chapter. In addition, the geometry of the
Pareto frontier and its relationship with the optimal weights for each objective is investi-
gated in Section 4.3.1. The extension of this approach to trace the points in the Pareto
frontier is discussed in Section 4.3.2, where it utilizes a preference-based approach to
find a specific point with the desired level of compromises between objective. A through
convergence analysis of the Pareto descent optimization with bilevel programming is
presented in Section 4.4.

Chapter 5  In this chapter, the notion of Pareto efficiency is defined as the fairness
problem in a supervised learning task. This notion is defined in Section 5.3, where some
well-defined and common notions of fairness are reduced to the notion of Pareto efficient
fairness. The empirical results of this framework are presented in Section 5.4, where
it clearly shows the proposed Pareto descent optimization approach outperforms other
state-of-the-art algorithms for the fairness-aware learning task.

Chapter 6  In this chapter, an extension of the Pareto descent optimization approach is
applied to an unsupervised task of dimension reduction. The formulation of the problem
in this setting is different from the one in supervised learning tasks, as well as the notion
of fairness. This formulation with the notion of Pareto fair subspaces is discussed in
Section 6.3. The optimization of this problem is presented in Section 6.5, where it is an
extension to the Pareto descent optimization, and its convergence analysis is discussed
in Section 6.5.2. The empirical results of applying this framework on two real-world
datasets are presented in Section 6.6, where it shows that the proposed Pareto descent
optimization outruns the other state-of-the-arts for this problem. Moreover, in this
section, it has been shown that using this optimization approach to achieve fair subspaces
for dimension reduction, can in fact boost the fairness of the downstream classification
tasks.
Chapter 7  In this chapter, some concluding remarks are discussed, as well as directions for future work, including but not limited to bias mitigation in federated learning, targeted data-driven approach for invariant risk minimization, and stochastic Pareto descent optimization.

Appendices  In Appendix A, the details of the framework that has been used to construct the ground truth dataset for bow echo classification task is discussed. This dataset has been used in Chapter 3 to show the efficacy of the targeted data-driven regularization approach. In Appendix B, the meta-data of datasets used in Chapters 5 and 6 are provided.

1.5.1 Notation

Throughout this dissertation, the following notations for mathematical statements has been adapted. We use bold face upper case letters such as $\mathbf{X}$ to denote matrices and bold face lower case to denote vectors such as $\mathbf{f}$. The Frobenius norm and trace of a matrix $\mathbf{X}$ are denoted by $\|X\|_F$ and $\text{tr} (\mathbf{X})$, respectively. The eigenvalues of a positive semi-definite matrix $\Sigma \in \mathbb{R}^{d \times d}$ are denoted by $\gamma_{\max}(\Sigma) = \gamma_1(\Sigma) \geq \gamma_2(\Sigma) \geq \ldots \geq \gamma_d(\Sigma) = \gamma_{\min}(\Sigma)$. The set of integers, $\{1, 2, \ldots, m\}$, is represented by $[m]$. 

Due to biases introduced by large real-world datasets, deviations of deep learning models from their expected behavior on out-of-distribution test data are worrisome. Especially when data come from imbalanced or heavy-tailed label distributions, or minority groups of a sensitive feature. Classical approaches to address these biases are mostly data- or application-dependent, hence are burdensome to tune. Some meta-learning approaches, on the other hand, aim to learn hyperparameters in the learning process using different objective functions on training and validation data. However, these methods suffer from high computational complexity and are not scalable to large datasets. In this chapter, a unified data-driven regularization approach is proposed to learn a generalizable model from biased data. The proposed framework, named as targeted data-driven regularization (TDR), is model- and dataset-agnostic, and employs a target dataset that resembles the desired nature of test data in order to guide the learning process in a coupled manner. The problem is cast as a bilevel optimization and propose an efficient stochastic gradient descent based method to solve it. The framework can be utilized to alleviate various types of biases in real-world applications. Empirically, on both synthetic and real-world datasets, the superior performance of TDR for resolving issues stem from these biases is shown.

2.1 Introduction

Drastically improving their performance, machine learning and, more distinctively, deep learning models, are becoming the main propulsion of technology in a variety of domains.
Notwithstanding their success, they are still suffering from different biases in the training data distribution. Biases, regardless of their nature, cause a mismatch between training and testing data distributions, which leads to a poor generalization of the model on the test data. A palpable form of these biases appears when the size of different classes or groups is unbalanced. When class sizes are not balanced, the imbalanced dataset problem arises [22, 71, 144], where large classes can dominate the training process, resulting in a model with low accuracy on small classes. A severe form of imbalanced dataset problem, appears in most real-world big datasets with immense number of classes, is long-tailed data distribution [30, 16, 122], where the distribution of classes is skewed. In this case, most of the data belongs to a few prevailing classes, while a huge number of classes are represented by a few number of samples. Another form of bias happens when the sample size of different groups in a categorical feature are unbalanced (e.g., gender). As a result, fairness issues would affect the training process, which engender a poor performance on minority groups in the dataset [64].

There are a variety of solutions introduced to alleviate the aforementioned issues caused by biases in training data. A generic idea behind these solutions is to adapt the training distribution to the desired properties of an unbiased dataset, whether it is accomplished either by resampling or assigning weights based on training loss to have a cost-sensitive weighting scheme [44, 92] such as boosting methods like AdaBoost [51] or curriculum learning [74]. However, relying merely on the training distribution using the main learning objective has shown to be not practically efficient [131].

A new strand of research is to augment single-objective learning models with additional data-driven constraints in order to alleviate the effect of bias in training data [58, 28, 131, 102, 74, 91]. The main idea behind constrained learning schema stems from the observation that the accuracy on training data is not a satisfactory criterion by itself, and should be accompanied by a data-driven regularization. An appealing data-driven regularization idea is to create a target dataset that resembles the desired properties of the unbiased distribution, and impose it to the training process as an additional constraint. A common practice of tuning hyperparameters at the end of training using a validation set is a simple example of this regularization, which is a cumbersome task. Some meta-learning approaches like [50], on the other hand, introduce a coupled framework to interlace the hyperparameter tuning using a validation set with the training process in order to guide it. [131] introduced an example reweighting scheme that attempts to address the bias issues in training distribution using a meta-learning framework. They exploit a validation set and use the model loss on that set to perturb the weight of each sample in the training
process borrowing ideas from [97].

While data regularized methods have shown to be successful in some applications, the computational burden of tuning hyperparameters limits their scalability to large datasets or training data with skewed classes. Moreover, these methods are usually application specific and lack a principled way to be generalized to different types of biases discussed before. To overcome these challenges, in this chapter, the targeted data-driven regularization framework is introduced, which employs a small target distribution, free of those biases in training data, and bilevel programming [27] to model the multi-objective structure on both training and target distributions. As a bilevel programming, TDR has two levels, one dealing with the main training process, while the other uses a well-tuned target dataset in order to optimize the weights of each desired class or group in the dataset. As it is elaborated later, the weight for each group is proportioned to the inner product of the gradients of target dataset and each group’s gradient vectors. Hence, as it is depicted in Figure 2.1, TDR, despite the standard training, learns the optimal weight for each group to avoid the majority group to prevail in finding the descent direction for gradient descent. The main contributions of this proposal are as follows:

- A unified framework is proposed for out-of-distribution generalization that exploits a small well-crafted target distribution to guide the learning process and cast it to a
bilevel programming problem to learn the optimal model. The framework is generic and can be utilized to tackle different types of biases and is free of hyperparameter tuning of other frameworks.

- The proposed framework addresses the computation and stability issues of previous frameworks, e.g. [131], while greatly improving the performance of the model.
- An efficient stochastic algorithm is proposed for TDR and it is validated via synthetic and real-world datasets on various problems.

2.2 Related Work

Two main challenges raised from biases in datasets are class imbalance and fairness problems. Contrary to fairness, studying imbalanced datasets is not a new problem in machine learning, hence there are a number of approaches to rectify this issue. Works related to the fairness problem are discussed in detail in Chapters 5 and 6.

Class Imbalance: Most of the approaches revolve around adopting the biased training distribution to the desired properties of an unbiased one, by either re-sampling or cost-sensitive mechanisms. In over-sampling, we add samples from minority classes to the dataset via either repetition or novel approaches like domain adaptation or synthetic data generation [6] and interpolation of neighboring samples [25]. Cost-sensitive learning, on the other hand, is the method of assigning weights to each sample’s loss based on the data distribution [77, 92]. It is common to choose weights as the inverse of class frequency for imbalanced datasets [71, 146]. Recent efforts use more intuitive approaches than inverse of class frequency for reweighting samples. For instance, in [30], based on the sample size of each class calculated the effective number of samples for that class, and balanced the loss based on that number. The proposed TDR framework for out-of-distribution generalization belongs to the latter group, where an unbiased and small target dataset is used to learn the weights of each group interleaved with the main training procedure. Having these weights trained during the main training task is the advantage of this framework over others.

Meta-learning: Apart from all these classical approaches, which are mostly burdensome to tune for each specific dataset or application, we can benefit from the new trend arising in machine learning domain, called meta-learning [4, 49]. While meta-learning
approaches are mostly used in few-shot and multi-task learning settings, they have also been adopted in hyperparameter optimization tasks [50] or addressing noise in data [73]. These frameworks aim at learning the parameters of the optimization on top of the main learning process using gradient descent. Following this trend, Ren et al. [131] proposed a new framework to address biases in the dataset, using a validation set, similar to test set and by using the perturbation idea of [97], they perturb each example’s weight in the training. Although its effectiveness is comparable to classical approaches, it encounters several issues, such as high computational complexity and instability of the perturbed weights, making it impractical in large-scale problems. Litany and Freedman [102] proposed a very similar framework to example reweighting for transfer learning. In TDR, a bilevel programming is utilized to develop a framework for addressing biases in the dataset, while avoiding the aforementioned issues. For a thorough discussion of existing methods refer to Section 2.4.

**Out-of-distribution Generalization:** Attracting much attention recently, various frameworks have been proposed for the problem of out-of-distribution generalization. Shankar et al. [137] suggest a combination of gradients on different domains to learn a model that generalizes well to new domains, and does not need domain signals like prior methods. Sun et al. [142] introduce a training approach during the inference time to adapt the model to new domains. Arjovsky et al. [7] try to make the model invariant to the spurious correlation in the feature data. The main advantage of TDR over these approaches is that it uses a small target dataset that can be adopted based on different applications, instead of using the whole dataset from other domains.

### 2.3 Targeted Data-driven Regularization

In this section, the targeted data-driven regularization framework is introduced and cast as a bilevel optimization problem. Then, an efficient two-level stochastic optimizer is proposed to learn the model.

#### 2.3.1 The proposed framework

Learning the learning process is a common practice nowadays in the machine learning field and in meta-learning approaches [4]. However, most existing studies focus on tuning hyperparameters in the learning process. It will be shown that utilizing a well-crafted small target dataset as a guidance for the main learning process can ameliorate the main
learning problem. This target dataset is much smaller than a normal validation dataset, and is well-picked to be free of those biases. In TDR, a different objective function can be imposed on the target distribution than the one for the main training distribution, depending on the learning problem that is being solved.

The way two objective functions and their parameters are defined is problem-independent. However, for the sake of exposition, problems such as imbalanced dataset and fairness in classification models are tackled here. These two types of problems have a common ingredient, i.e., unbalanced distribution of different categories in the dataset. If this unbalanced nature of the data happens in the labels that data points are going to be classified based on them, the problem is class-imbalanced dataset. But if one of the sensitive categorical features in the dataset (e.g., gender) happens to be unbalanced, the issue of fairness would raise. In both scenarios, a prediction problem is being solved on dataset $\mathcal{T}$ with $n$ training samples, from input space $\mathcal{X}$ to label domain $\mathcal{Y}$, where each sample point is defined as $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. The notation $g_i = \ell(\theta; (x_i, y_i))$ is used for sample loss, where $\ell(\cdot; \cdot)$ is the training loss function and $\theta$ is the parameter set of the model, to denote the training loss on $i$th sample $(x_i, y_i) \in \mathcal{T}$.

In order to address aforementioned biases, we need to penalize the loss of samples from different classes or groups with different weights, hence, a weight vector is defined, $w \in \mathbb{R}^c_+$, where $c$ is the number groups of the discriminatory feature (i.e., label in imbalanced data or sensitive feature in fair learning). Let $D \in \{0, 1\}^{n \times c}$ denote the assignments of $n$ training samples to $c$ groups of the discriminatory feature. For a model parameter $\theta$ and a fixed weight vector $w$, the loss over training examples is defined as:

$$G(w, \theta; \mathcal{T}) = (Dw)^\top g, \tag{2.1}$$

where $g = [g_1, \ldots, g_n]^\top$ is the vector of losses per sample, and $G(., .; .)$ is the weighted loss over training samples.

Equipped with (2.1) as training goal, for a known weight vector $w$ the optimal parameters $\theta$ can be found by minimizing the objective. However, the samples in the target dataset are used to adaptively learn the optimal weight vector and guide the training process. To this end, the loss over a small unbiased target dataset $\mathcal{V}$ is defined as:

$$F(w, \theta^*(w); \mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{(x_i, y_i) \in \mathcal{V}} f(\theta^*(w); (x_i, y_i)), \tag{2.2}$$

where $|\mathcal{V}|$ is the number of samples in $\mathcal{V}$, $\theta^*(w)$ is the minimizer of the loss function in (2.1), and $f(\cdot; \cdot)$ is the target loss function which may or may not be same as training.
loss $\ell(\cdot; \cdot)$. It is worth noting that target dataset could be a part of training dataset or it could be separated from it, similar to a validation dataset.

Having two interlaced optimization problems, the TDR can be cast as a game between two players, called leader and follower [13]. Both players want to minimize their specific objective functions which results in the following bilevel programming:

$$
\mathbf{w}^* \in \arg\min_{\mathbf{w}} \ F(\mathbf{w}, \theta^*(\mathbf{w}); V)
$$

\text{s.t.} \quad \theta^*(\mathbf{w}) \in \arg\min_{\theta} \ G(\mathbf{w}, \theta; T),

(2.3)

where the objective of the leader and the follower, $F(\cdot; \cdot)$ and $G(\cdot; \cdot)$, are defined in (2.2) and (2.1), respectively.

It is worth noting that despite the similarity of this algorithm with a universal bilevel programming, the two levels are being optimized on different data distributions. This is the key difference that makes the TDR framework capable of solving some challenging problems with a data-driven regularization using target distribution.

Also, unlike constrained optimization problems, where first-order algorithms such as gradient descent or stochastic gradient descent can be easily utilized, in the above formulation, to find the optimal $\mathbf{w}$, we need to first solve the inner objective $G(\cdot; \cdot)$ with respect to $\theta$ which depends on $\mathbf{w}$. As a result, any iteration of GD algorithm requires fully optimizing the inner optimization which makes the optimization intrinsically hard. In next section, an efficient stochastic optimization algorithm is proposed to approximately solve the stated bilevel optimization problem.

### 2.3.2 Stochastic bilevel optimizer

In general, if we want to solve a bilevel program, we have to solve each level to reach a local minimum. However, in order to control the computational complexity of the algorithm, an stochastic method is proposed where it iteratively updates the parameters of the outer optimization problem. In this setting, instead of solving the inner problem completely per iteration, it only takes few gradient steps over the parameters of each. These loops are continued until it converges or stops early at the best solution. This requires to define a number for inner iterations $t_{in}$, which indicates how many stochastic steps is required to be taken each time the algorithm is in the inner problem. Moreover, similar to [56], $t_{in}$ could be tuned adaptively to a monotonically increasing sequence, since in the beginning the solution might be far away from a local minimum of the inner problem. As it moves along in the training process, the solution is getting closer to
Algorithm 1: Targeted Data-driven Regularization

```
input $\hat{\theta}_0 \in \mathbb{R}^d, w_0 \in \mathbb{R}^c, \eta_{in}, \eta_{out}, t_{in}$
for $k = 0, 1, \ldots$ do
    set: $\theta_0 = \hat{\theta}_k$
    for $m = 0, 1, \ldots, t_{in} - 1$ do
        $\theta_{m+1} = \theta_m - \eta_{in} \cdot \nabla_\theta \tilde{g}(w_k, \theta)$
    end
    set: $\hat{\theta}_k = \theta_{t_{in}}$
    $w_{k+1} = w_k - \eta_{out} \cdot \nabla_w \tilde{f}(w, \hat{\theta}_k)$
end
return $\hat{\theta}_k$
```

The local minimum of the inner problem, hence, more stochastic steps might be needed toward it before updating weights.

The inner problem in TDR is minimizing the objective function of the main problem, defined in (2.1), with respect to its parameter set, $\theta$, on the training data distribution $\mathcal{T}$. Using mini-batch gradient descent, we want to minimize the inner problem as follows:

$$\theta^*(w) \in \arg \min_{\theta \in \Theta} \{ \tilde{g}(w, \theta) = \mathbb{E}_{\xi \sim \mathcal{T}} [G(w, \theta; \xi)] \} , \quad (2.4)$$

where $\Theta$ is the domain of all feasible solutions for $\theta$, $\xi$ is a mini-batch drawn from the training data distribution $\mathcal{T}$ with the mini-batch size of $|\xi| = b$ and $\tilde{g}(\cdot)$ denotes the empirical loss over $\xi$. Then the outer problem gets the solution of the inner problem and minimizes its own objective function, defined in (2.2), over its parameter set, $w$, using data samples of the target dataset $\mathcal{V}$:

$$w^* \in \arg \min_{w \in \Omega} \{ \tilde{f}(w, \theta^*(w)) = \mathbb{E}_{\xi_v \sim \mathcal{V}} [F(w, \theta^*(w); \xi_v)] \} , \quad (2.5)$$

where $\Omega$ is the feasible domain for $w$, and $\xi_v$ is a mini-batch drawn from the target distribution with the size $|\xi_v| = b_v$ and $\tilde{f}(\cdot)$ is the loss over $\xi_v$. The key point that connects two levels together is the solution of the inner level, which is a function of $w$.

The proposed generic TDR framework is defined in Algorithm 1, in which the procedure is illustrated in Figure 2.2.

For both aforementioned use cases of TDR, namely, imbalanced dataset and fairness issues, a discriminatory feature vector is needed. In imbalanced dataset scenario, this vector is the label of samples. In fairness problems, generally, there is an unbalanced categorical feature, which turns to an unfair classifier in favor of the majority category of
that feature in the dataset. As an example, when in a dataset the numbers of samples for different genders are unbalanced, the resulting class is inclined to have a better performance in terms of accuracy for the majority category rather than the minority group. In this case, the discriminatory vector is a feature vector from the data itself, where the goal is to find the optimal weight for each category in the classification. The weight for each category can be seen as a sampling probability, where in a simple stochastic GD it is considered to be a uniform distribution, while in TDR, it can find the optimal probability distribution for categories. Thus, to make these weights resemble a probability distribution, we have to impose two constraints on the weights. The weights for each sample in the mini-batch in step $k$ is $\tilde{w}_k \in \mathbb{R}^{b \times 1}$, and it is defined as:

$$
\begin{align*}
    w_k^+ &= \max(0, w_k), \\
    \tilde{w}_k &= \frac{Dw_k^+}{\|Dw_k^+\|_1},
\end{align*}
$$

(2.6) (2.7)

where $w_k$ comes from step $k$ in Algorithm 1, and $\bar{D}$ is the stochastic counterpart of the discrimination matrix for mini-batch $\xi_k$. Each sample’s weight in the mini-batch should be positive and also sum over all samples’ weights in the mini-batch should be equal to one. Hence, the weights of samples in the mini-batch are normalized with their $l_1$ norm to be in range of $[0, 1]$ and sums up to 1.

The training framework defined in Algorithm 1 is straightforward and can be implemented using any machine learning or deep learning libraries such as Tensorflow [1].
and PyTorch [124]. The main challenge is the connection between inner and outer models. Because the outer objective function is an implicit function of the weights used in the inner level, this dependency should be created in the model construction time so the derivative of the outer objective function with respect to the weights could be algorithmically computable. For instance, in Tensorflow using Estimator API, after construction of the inner model’s graph, the outer model’s graph can be created using

$$\theta_{\text{out}} = \theta_{\text{in}} - \eta_{\text{in}} \nabla_{\theta} \tilde{g}(w, \theta),$$

which is simply the update rule of the inner problem. Thus, the parameters of the outer model would be an implicit function of weights.

### 2.3.3 Weights Interpretation

To better understand the learned weights for each class or group during the training, it is beneficial to further elaborate on the update rules of Algorithm 1. First, in the inner loop, it updates the parameters of the model by propagating weighted gradients of $t_{\text{in}}$ mini-batches. Thus, the update rule of the inner level at the step $k$ of the outer level can be rewritten as:

$$\theta_{m+1} = \theta_{m} - \eta_{\text{in}} \cdot \nabla_{\theta} \tilde{g}(w_{k}, \theta)$$

$$= \theta_{m} - \eta_{\text{in}} \cdot \frac{1}{|\xi|} \sum_{i=1}^{[\xi]} w_{k}^{c_{i}} \cdot \frac{\partial g_{i}(\theta)}{\partial \theta} \bigg|_{\theta=\hat{\theta}_{m}},$$

(2.8)

where $w_{k}^{c_{i}}$ is the weight of the $i$th sample in the mini-batch, which belongs to class or group $c_{i}$. Now, the update rule of the outer level can be written for weights of each class or group separately as follows:

$$w_{k+1}^{c} = w_{k}^{c} - \eta_{\text{out}} \cdot \left[ \frac{\partial f_{j}(w, \hat{\theta}_{k})}{\partial w} \right]_{c}$$

$$= w_{k}^{c} - \frac{\eta_{\text{out}}}{|\xi_{v}|} \sum_{j=1}^{[\xi_{v}]} \frac{\partial f_{j}(\theta(w))}{\partial w^{c}} \bigg|_{w=w_{k}}$$

$$= w_{k}^{c} - \frac{\eta_{\text{out}}}{|\xi_{v}|} \sum_{j=1}^{[\xi_{v}]} \left( \frac{\partial f_{j}(\theta(w))}{\partial \theta} \right)^{\top} \left( \frac{\partial \theta(w)}{\partial w^{c}} \right)$$

$$= w_{k}^{c} + \frac{\eta_{\text{out}}\eta_{\text{in}}}{|\xi_{v}| |\xi|} \cdot \left( \sum_{j=1}^{[\xi_{v}]} \frac{\partial f_{j}(\theta(w))}{\partial \theta} \right)^{\top} \left( \sum_{i=1}^{[\xi]} \frac{\partial g_{i}(\theta)}{\partial \theta} \right),$$

(2.9)

where $f_{j}(\cdot)$ is the empirical risk of the $j$th sample of the mini-batch drawn from the target dataset and $[\cdot]_{c}$ is the $c$th element of the vector.

---

1The code repository https://github.com/mmkamani7/Targeted-Meta-Learning
The final equality reveals an elegant property about the learned weights in the TDR framework. Indeed, it demonstrates that the weight of each class or group changes with the conformity of the average gradients of samples from that class or group with the average gradients of the target dataset. As long as the average gradients of one group is aligned with the average gradient of the target dataset, its weight will increase; otherwise, it will decrease. In this case, when the average gradients of a majority group is not highly correlated with the average gradients of the target dataset, they fail to dominate the descent direction for updating the parameters of the network.

### 2.4 Time Complexity Comparison

In this section, the computational complexities of other approaches that use a data-driven regularization scheme, closely related to TDR, are compared. Also, it is shown that they can be considered as a special case of the proposed TDR framework. These frameworks are example reweighting [131], SOSELETO [102], MentorNet [74], and Deep Bilevel learning [73]. In all of these frameworks, except for deep bilevel learning, the main idea is to have different weights in the cost function for different samples, reflecting samples’ importance in each of the aforementioned tasks. Hence, the computational complexity of these approaches is at least $t_{in}$ times the computational complexity of the TDR. Deep bilevel leaning is similar to TDR, except for their approximation for inner level, which makes it faster at the price of degradation in accuracy.

Example reweighting [131] uses the perturbation idea [97], where they investigate the effect of input perturbation on the output of a network. They try to address the problem of an imbalanced dataset using a balanced validation set, by perturbing the weight of each sample from zero. The main difference between example reweighting and TDR is that they perturb each samples’ weight from zero using the gradient direction, rather than learning the weights with gradient descent approach. Hence, each time the network encounters a specific example, without considering its weight in last epoch, they assign a new weight to it using perturbation. In addition, due to sample-based nature of the algorithm, as indicated in that paper, the computational complexity of perturbing weights is linear in the number of samples, while in TDR it is linear in number of classes, which is a significant improvement. In practice, in Chapter 3, it is shown that TDR outperforms example reweighting in both speed and accuracy. Their approach can be rewritten as a bilevel programming similar to Algorithm 1, where $t_{in}$ is 1, which means every level updates only once before going to another level. This is equal to solving the
bilevel programming as a Lagrangian form and finding the solution for primal and dual variables in an iterative fashion. Because of this high computational cost, applying this method to rather large networks such as ResNet20 is not feasible, compared to TDR.

SOSELETO introduces a framework for transfer learning using a target dataset [102]. In this framework, they consider two models for source and target datasets, and for each model, parameters considered as a union of two parts, namely, feature part and classification part. In both target and source models, the feature part of the parameters is the same. They optimize the classification part’s parameters for the target model, using a weighted loss on the source data. Hence, learning weights for source data points and feature’s part parameters for target model at the same time can be written as a bilevel program. If we combine the two parameter sets together and have a single parameter set, the framework is exactly similar to example reweighting framework with its aforementioned issues.

Mentornet classifies a dataset with noisy labels using the curriculum learning approach [17]. They train another network, with features extracted from the main network as its input, to learn the weight of each sample in the training operation. Similar to example reweighting, the objective of this framework can be written as a Lagrangian problem. It is worth noting that in this problem, the parameters of the main network or student net are not involved directly, but they are affected by changing the weights for each sample calculated from the simple network of $g(\cdot)$. The main difference between this framework and the previous ones is that the weights of each samples are output of a network, which should be trained (based on the optimal weights, 1 if the main label is correct and 0 otherwise). However, in previous ones, we consider the samples’ weights as a hyperparameter. Based on the optimization problem in (2.3), this problem can be rewritten as a bilevel problem with equal inner and outer functions.

While deep bilevel by [73] is closely related to TDR’s idea in order to cast the problem as a bilevel optimization, the idea of bilevel programming for hyperparameter tuning is not new (e.g. [50]). Further, their idea of having different weights for different batches and trying to learn those weights based on gradients on the validation set is more related to [131]. Despite the relatedness among these papers, TDR’s idea is different in three key aspects as follows:

- The main idea of TDR is to employ a target dataset that is free of those mentioned biases as a surrogate to not well-defined objectives in the training (when the training and test distributions are different). In deep bilevel, they randomly choose both training and validation datasets, hence, their framework is not designed to address
problems such as imbalanced datasets. Further, their validation dataset is changing in each epoch and is relatively big in size, which is more like a cross-validation. However, in TDR the target dataset is very small \( |V| \leq 0.0002 \), it could be part of the training itself, and is not changing during the training, which indicates the capability of TDR.

- The optimization approaches are completely different. They approximate the inner level with Taylor expansion, which makes it a quadratic function to reduce the computational complexity of the bilevel problem by avoiding the second order derivatives. TDR is approximating the bilevel programming with a stochastic version of it, in which the computational complexity depends on \( t_{in} \).

- Their experiments are mainly focused on noise in data and labels, like other frameworks mentioned in this chapter, such as [131, 74]. In most of these methods, the key property is that noisy data usually have higher loss, hence they should be ignored or weighted down to avoid their dominance in the training. The proposed framework is more general because it is not bounded to a specific problem and can adapt to different problems by adapting the target dataset, like using a small clean dataset as the target dataset for noisy training data.
This chapter is a showcase of the proposed TDR framework in Chapter 2 for the class-imbalance problem as the source of bias in the learning task. In order to show the efficacy of this algorithm, it has been applied to both synthetic and real-world datasets with different levels of imbalance between classes. In what follows, the results of applying this algorithm on various datasets and models are presented.

### 3.1 Experimental Results

In this section, the experimental results for the targeted data-driven regularization (TDR) algorithm is presented to show its efficacy in dealing with imbalanced datasets. To do so, in the first step the MNIST\(^1\) and CIFAR10\(^2\) datasets are used to create synthetic imbalanced datasets. In addition to these synthetic imbalanced datasets, TDR is being applied to a real-world dataset for severe weather prediction. This dataset consists of radar images of the whole US continent using data gathered from NEXRAD level III radars (or WSR-88D)\(^3\) of the National Weather Services.

As stated in Section 2.3, in an imbalanced dataset scenario, the discriminatory vector is the label vector in the dataset. Hence, in this case for each class a weight is learned and it gets updated based on the gradient descent direction of the target objective function. In order to better measure the imbalance level in a dataset, an *imbalance ratio* \((\rho)\) is introduced, which is the ratio between the number of samples in the smallest class \((n_m)\)

\(^1\)http://yann.lecun.com/exdb/mnist/
\(^2\)https://www.cs.toronto.edu/~kriz/cifar.html
\(^3\)https://mesonet.agron.iastate.edu/archive/
and the largest class \((n_M)\), i.e., \(\rho \equiv \frac{n_m}{n_M}\). Using TDR, models are trained on imbalanced and long-tailed data distributions with a small balanced target distribution. In order to demonstrate the robustness of the proposed algorithm, it is compared to example reweighting [131]. In this experiment, the parameters are chosen as: \(b = 50\), \(b_v = 10\), \(\eta_{in} = 0.001\), and \(\eta_{out} = 0.2\).

3.2 Synthetic data

In this section, TDR is being applied to synthetically imbalanced datasets using two common vision datasets, MNIST and CIFAR10.

3.2.1 MNIST Dataset

To show the effectiveness of TDR, first, it is applied to the MNIST dataset. Using that, two different datasets are generated, one with only two classes, and the other using all ten classes in the dataset. For the first experiment, an imbalanced dataset is created using only two digits images. To better examine the framework, the two most confused digits are used based on the MNIST classification confusion matrix \(^4\), which are ‘4’ and ‘9’. For this experiment the class 4 is the minority class and 9 is the majority class. If the total number of samples in training is \(N\), the size of minority class and that of majority class are \(\frac{\rho N}{\rho + 1}\) and \(\frac{N}{\rho + 1}\), respectively. The LeNet [98] model is used as the base model with cross entropy over output logits as the loss function. This is a simple convolutional neural network with two convolution layers each with a max pooling at the output following with 3 fully connected layers. For the first experiment, imbalance ratios from \(\rho \in \{0.1, 0.05, 0.02, 0.005\}\) are chosen and different imbalanced datasets are generated for \(N = 5,000\). For instance, \(\rho = 0.005\) means that for every 200 samples from the larger class, only 1 sample is from the smaller class. The target distribution is a dataset with 50 data points balanced over different classes. The test dataset is a balanced set with 2,000 samples. Figure 3.1 shows the performance of TDR and sample reweight on the aforementioned imbalanced dataset with ratio of \(\rho = 0.005\), based on epochs (Figure 3.1(a)) and wall clock time (Figure 3.1(b)) of training. It can be inferred that the TDR outperforms example reweighting on accuracy with a substantially higher convergence speed.

In addition to accuracy on balanced test set, in classification of an imbalanced dataset,

\(^4\)https://ml4a.github.io/demos/confusion_mnist/
Figure 3.1: Test accuracy on imbalanced dataset containing two classes (‘4’ and ‘9’) of MNIST. The test dataset is a balanced dataset of the two classes. Total number of training samples is 5,000 and class 4 is in the smaller class with the imbalance ratio of $\rho = 0.005$. Target distribution has 50 balanced samples. Targeted meta learning outperforms example reweighting, both in terms of accuracy and speed.

Figure 3.2: Recall rate of the smaller class 4 during the training of the experiment in Figure 3.1 with imbalance ratio $\rho = 0.005$. Similar to test accuracy, TDR outruns example reweight in recall rate.

The crucial metric is the recall rate, which shows the performance of the classifier on the positive (smaller) class data points. Figure 3.2 shows the recall rate of the class 4 in training procedure, in which the superiority of the TDR is noticeable.

Figure 3.3 presents a comparison of these two frameworks on different imbalance ratios. To compare reliably, the training procedure is repeated for 5 times and the
Figure 3.3: Comparing the test accuracy of TDR with example reweighting for different imbalance ratios from $\rho \in \{0.1, 0.05, 0.02, 0.005\}$. The dataset is MNIST with two classes (4 and 9), with the same setting as in Figure 3.1. Each experiment is repeated 5 times to examine the robustness of algorithms.

average and standard deviation of the test accuracy are reported. TDR not only boosts the performance, but also is more stable compared to example reweighting. This can be the result of incremental class weight updates in TDR, compared to perturbing each sample’s weight for each iteration in the example reweighting.

As for the second experiment, the same model is used, however, this time a long-tailed dataset is created. To generate a long-tailed dataset from MNIST, using the same scheme as [30], we can set $n_i = n_0 \mu^i$, in order to decrease the class size exponentially, where $n_i$ is the number of samples in class $i$. Hence, based on the definition, the imbalance ratio would be $\rho = \frac{n_0}{n_i} = \mu^9$. The same imbalance ratio as before is used from $\rho \in \{0.1, 0.05, 0.02, 0.005\}$, and both TDR and sample reweighting are tested on these datasets. Figure 3.4(a) shows the result of this experiment on two frameworks. As it can be implied, example reweighting has an extremely poor performance on long-tailed dataset. This is due to perturbing weights for each sample at each iteration, which can introduce a huge noise to the training. Figure 3.4(b) exhibits the distribution of learned weights for each class in the training process. This observation shows that, the optimal weights for each class would not necessarily correlate with its inverse of frequency.
Figure 3.4: (a) The test accuracy of TDR vs. example reweighting for different imbalance ratios $\rho \in \{0.1, 0.05, 0.02, 0.005\}$ with long-tailed MNIST dataset. Each experiment is repeated 5 times. (b) Distribution of learned weights for each class for $\rho = 0.02$. Each row shows the distribution of weights for each class during training. The size of each class is decreased exponentially, however, the learned weights are not exactly proportional to the inverse of their size.

### 3.2.2 CIFAR10 Dataset

Similar to MNIST experiment in Figure 3.1 and Figure 3.3, same experiments are run on CIFAR10 Dataset with having the same structure for creating imbalanced dataset of classes “4” and “9” with class “4” as the minority class in the dataset. The experiments are run for 4 different imbalance ratios from $\rho \in \{0.3, 0.2, 0.1, 0.05\}$. Also, in this experiment, two different network architectures are chosen, ResNet20 [65] and a simple 4-layer ConvNet with 2 fully connected layers at the top for classification. In order to show the efficacy of TDR framework it is compared with example reweighting [131], hard weighting with inverse of class frequency, random weighting, and without any weights. The results for the ConvNet on datasets with different imbalance ratio are depicted in Figure 3.5, where the superiority of TDR is noticeable. Example reweighting is better than random weights, but cannot even beat the learning without any weights, nor the hard weighting with inverse of frequency. The other major issue of example reweighting, is the high computational complexity, which prevents us from running it on larger networks like ResNet. The detailed results of this experiment on both models are in Table 3.1, in which TDR has the highest test accuracy on all the experiments.

In addition to two classes, similar to the long-tailed MNIST experiment, an experiment
Figure 3.5: Comparing the test accuracy of TDR with example reweighting, hard weighting with inverse of frequency, random weights and no reweighting. These experiments are run for different imbalance ratios from $\rho \in \{0.3, 0.2, 0.1, 0.05\}$. The dataset is CIFAR10 with two classes (4 and 9) and the model is a 4-layer ConvNet. Each experiment is repeated 5 times to examine the robustness of algorithms. The detailed results are in Table 3.1

<table>
<thead>
<tr>
<th>Imbalance Ratio</th>
<th>0.3</th>
<th>0.2</th>
<th>0.1</th>
<th>0.05</th>
<th>0.3</th>
<th>0.2</th>
<th>0.1</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDR</td>
<td>0.9011</td>
<td>0.9222</td>
<td>0.8878</td>
<td>0.8756</td>
<td>0.9284</td>
<td>0.9266</td>
<td>0.9175</td>
<td>0.9025</td>
</tr>
<tr>
<td></td>
<td>±0.018</td>
<td>±0.005</td>
<td>±0.049</td>
<td>±0.014</td>
<td>±0.004</td>
<td>±0.002</td>
<td>±0.002</td>
<td>±0.003</td>
</tr>
<tr>
<td>Example Reweighting [131]</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0.4999</td>
<td>0.737</td>
<td>0.5977</td>
<td>0.6641</td>
</tr>
<tr>
<td></td>
<td>±0.001</td>
<td>±0.034</td>
<td>±0.067</td>
<td>±0.155</td>
<td>±0.019</td>
<td>±0.021</td>
<td>±0.015</td>
<td>±0.006</td>
</tr>
<tr>
<td>Hard Weighting</td>
<td>0.8545</td>
<td>0.8966</td>
<td>0.8498</td>
<td>0.7969</td>
<td>0.9007</td>
<td>0.8672</td>
<td>0.9055</td>
<td>0.8899</td>
</tr>
<tr>
<td></td>
<td>±0.002</td>
<td>±0.146</td>
<td>±0.002</td>
<td>±0.002</td>
<td>±0.147</td>
<td>±0.074</td>
<td>±0.002</td>
<td>±0.001</td>
</tr>
<tr>
<td>Random Weights</td>
<td>0.4998</td>
<td>0.5094</td>
<td>0.5005</td>
<td>0.5006</td>
<td>0.5656</td>
<td>0.5317</td>
<td>0.4978</td>
<td>0.499</td>
</tr>
<tr>
<td></td>
<td>±0.015</td>
<td>±0.114</td>
<td>±0.108</td>
<td>±0.108</td>
<td>±0.144</td>
<td>±0.145</td>
<td>±0.166</td>
<td>±0.002</td>
</tr>
</tbody>
</table>

Table 3.1: Experiment results on imbalanced CIFAR10 dataset with two classes of “4” and “9” (“4” is the minority class). Two different models are used, namely, ResNet20 and a simple 4-layer ConvNet with 2 fully connected layers as the classifier. TDR is compared with example reweighting, hard weights with inverse of class frequency and no weights or standard training.

is run on CIFAR10 data. In order to make the data long-tailed, the size of classes is decreased with $n_i = n_0 \mu^i$ and two datasets are generated with imbalance ratios of $\rho \in \{0.1, 0.2\}$. Then, the TDR framework is run on the datasets, as well as example reweighting, hard weighting with inverse of class frequency, random weighting, standard training, on two models of ResNet20 and the similar 4-layer ConvNet. Because of the aforementioned reasons, running example reweighting on ResNet20 is not computationally
feasible for the sake of comparison. The results of these experiments on Table 3.2 indicates the superiority of TDR for long-tailed CIFAR10 on ResNet20. Although ConvNet is not a suitable model choice for this training, it is included in the results for a better comparison, where standard training beats all other weighting schemes.

<table>
<thead>
<tr>
<th>Imbalance Ratio</th>
<th>ResNet20</th>
<th>ConvNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>TDR</td>
<td>0.5602 ± 0.017</td>
<td>0.5758 ± 0.046</td>
</tr>
<tr>
<td>Example Reweighting [131]</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Hard Weighting</td>
<td>0.367 ± 0.058</td>
<td>0.3336 ± 0.077</td>
</tr>
<tr>
<td>Random Weights</td>
<td>0.0908 ± 0.015</td>
<td>0.1062 ± 0.076</td>
</tr>
<tr>
<td>No Reweighting</td>
<td>0.3628 ± 0.08</td>
<td>0.3308 ± 0.066</td>
</tr>
</tbody>
</table>

Table 3.2: Experimental results on long-tailed CIFAR10, with imbalance ratio from $\rho \in \{0.1, 0.2\}$ with two models, ResNet20 and ConvNet. We can use $ni = n0 \mu_i$ to decrease the class size and make the dataset long-tail. TDR achieves the best accuracies in ResNet20, however the training without any weights is doing better in ConvNet, which is not a suitable model for training of this dataset.

### 3.3 Severe weather prediction dataset

In real-world applications, the primary challenge is often to detect critical incidents in datasets. Notwithstanding their importance in the classification tasks, normally those critical incidents are scarce in the dataset. Therefore, based on earlier discussions, a typical classifier would fail miserably in detecting these incidents.

One of the conspicuous examples of these critical incidents is severe weather detection using radar, satellite, and other sensor data. Sever weather conditions such as tornadoes, thunderstorms, and straight-line winds, are occasional phenomena, but can be spotted in radar or satellite images with some specific patterns. One of these patterns, associated with some severe weather conditions such as thunderstorms and straight-line winds, is called Bow Echo, because it has archer’s bow shape pattern in radar images as it is depicted in Figure 3.6. Detecting and predicting the formation of bow echoes could help meteorologists predict related severe weather conditions and prevent their detrimental
In this experiment NEXRAD level III radar data is used in order to create a dataset of radar images for a whole year of 2008 gathered from 160 high resolution radars across the United States. The model is being tested on a balanced set of bow echo and non-bow echo samples from the year 2009. The year 2008 is chosen for training because of a high number of severe weather activities in that year. The reflectivity images of radar data is being used, similar to the Figure 3.6, which are 4-bit color map with spatial size of $2,600 \times 6,000$ pixels. The colors in these images, as it is indicated in Figure 3.6, are showing different amplitude of the reflected signal in dBZ from 0 to 75 dBZ. Bow echo patterns mostly revealed in areas with more than 50 dBZ, i.e., in areas with orange or red colors in the images. These images are created from radar data every 5 minutes, hence, there are 288 images per day, which can lead to more than $105K$ images every year. Despite the huge number of images each year, number of images with a bow echo sample on it is very limited. For instance, in the year 2008 there are only 1,821 images from 81 different instances that are labeled as bow echo samples. Therefore, this dataset, similar to other severe weather detection and prediction datasets, is greatly imbalanced with $\rho = 0.017$. The data distribution is immensely skewed toward normal data points, as it is the case for most of related critical incident detection applications. Thus, the TDR framework is applied to this dataset to overcome the imbalance problem in this case. The detailed framework for creating the ground truth bow echo datasets of these two years (2008 and 2009) are discussed in Appendix A, where morphological steps are used to detect these patterns while it requires careful preprocessing steps.

For this dataset, TDR is applied on ResNet20 [65] model, with image size of $52 \times 180$. The target distribution is a balanced dataset of both bow echo and non-bow echo samples.
Figure 3.7: Accuracy, recall, and precision rates on balanced test dataset after 11 epochs of training for targeted-meta learning, hard reweighting with inverse of frequency, and without weights. The training dataset contains the complete radar images from year 2008 with \( \rho = 0.017 \), while the test set is a balanced dataset of bow echo and non bow echo samples from year 2009. Test accuracy and recall rate on bow echo samples reach to 0.8605 and 0.855 respectively. With hard weights the recall rate increases but at the cost of decreasing precision.

from year 2008 with size of \( |\xi_v| = 273 \) that has 137 bow echo samples. The balanced test set contains 3,524 images from year 2009, which has 1,762 bow echo samples. In this experiment, the parameters of \( \text{TDR} \) are set to \( b = 50 \), \( b_v = 10 \), \( \eta_{in} = 0.001 \), and \( \eta_{out} = 0.2 \). For comparison, the results for hard weighting classes with inverse of their frequency and also standard training without weights are added. The result of this training after 11 epochs in Figure 3.7 reveals that \( \text{TDR} \) has an exceptional capacity on addressing biases problem in this dataset, by achieving more than 86\% in accuracy and 85\% in bow echo recall rate.

32
As it was alluded to in previous chapters, dealing with bias in data in a learning problem requires considering multiple objectives, in addition to the main learning objective. The primary goal of these additional objectives is to diminish the consequences of biases on those underrepresented or misrepresented groups or class in the dataset during the training procedure. These objectives will prevent the learning procedure to drift toward a biased model and tries to recover an unbiased model we can achieve in the absence of the bias.

In the targeted data-driven approach, introduced in Chapter 2, a novel way to include these additional objectives in the training was proposed and examined. The efficacy of this approach in practice, presented in Chapter 3, demonstrates that using additional objectives we are able to recover the unbiased model from the biased data. Despite these merits, this approach requires to have access to a tiny target dataset that is free of those biases, which might not be practical in some applications. For instance, in the case of the fairness problem in the machine learning models, it is suggested that we are dealing with two sources of bias in the datasets: under-representation and labeling bias [18]. The under-representation case is similar to the class-imbalance problem, where here we have smaller samples for underrepresented groups (e.g. female applicants in STEM programs), which led the model to be biased toward the majority group. The labeling bias, on the other hand, comes from corrupted data to benefit one group or harm another, such as in systemic racism that produces biased ground truth data for algorithmic decision-making systems to be learned upon. The former type of bias could be addressed using the targeted data-driven algorithm, however, for the latter obtaining
a target dataset does not seem to be practical.

Hence, in this chapter, we pursue a more generic approach in bias mitigation that does not depend on a target dataset and can be applied to different forms of biases. Considering the fact that the bias mitigation problem is dealing with multiple objectives in nature, a conspicuous approach is to formulate it as a vector minimization or multiobjective optimization problem [112, 66, 31]. In a scalar optimization problem, the goal is to minimize the objective function until we reach a local or global minimum, where with a small change in the model the objective function will not decrease any more. However, when we are dealing with multiple objectives, it is most probable that their minimizers are not coinciding in the parameter space, meaning, minimizing an objective might hurt other objectives. Thus, there is trade-off between objectives that ought to be minimized, and the ultimate goal of multiobjective optimization is to find the points with optimal compromises between objectives.

The optimal trade-off or compromise in this setting reflects on points that there is no other point in the feasible set with equal or better objective values\(^1\). These optimal trade-off points are called Pareto Efficient, which are named after Vilfredo Pareto\(^2\) an Italian economist and engineer who introduced this notion in his works on economic efficiency and income distribution. As can be inferred, these points are not unique, and depends on the level of compromises for each objective we can find a different solution. A set of all Pareto efficient points in a vector optimization problem is called Pareto Frontier. Therefore, there are two sets of questions required to be answered in multiobjective optimization:

1. How can we find a Pareto efficient point for a multi-criteria optimization problem?

2. How can we find different points on the Pareto frontier with different levels of compromises for each objective?

There are a vast number of studies and proposals to answer each of these questions. For the first question, the solutions can be categorized into two groups of heuristic and analytic approaches. Analytical approaches include but are not limited to different scalarization approaches such as constrained optimization, Chebyshev distance, and minmax optimization. Despite the variety of solutions, there are not many algorithmic approaches that can be applied to the vast majority of learning problems with theoretical guarantees of convergence to a Pareto efficient point. The second question is a much

\(^1\)The formal definition is provided in Definition 2
\(^2\)https://en.wikipedia.org/wiki/Vilfredo_Pareto
greater challenge than the first one since it requires finding the geometrical structure of
the Pareto frontier to be able to trace the Pareto efficient points along its surface. There
have been various efforts to address this problem, but this is still a challenging and open
problem to be solved.

In this chapter, the questions are answered by providing two algorithmic solutions.
For the first question, we propose a bilevel Pareto Descent Optimization algorithm that
can be utilized by any classification model— which is trainable via gradient descent,
to generate Pareto efficient models. The bilevel structure proposed for solving the
multiobjective optimization not only is novel from the methodology perspective but also
grants us an efficient tool for convergence analysis of these optimization problems. This
structure can pave the way for the convergence analysis of the stochastic multiobjective
optimization in future work. Using Pareto descent optimization, we propose a novel
first-order algorithm to trace the points on the Pareto frontier, which can be generalized
to any other multi-criteria optimization problem and does not suffer from the limitations
of previous approaches. In the next two chapters, we apply these algorithms to fairness-
aware learning problems to show how we can mitigate the effects of biases using these
proposed algorithms.

4.1 Pareto Efficiency

In a typical supervised learning our goal is to learn a function \( f : \mathcal{X} \mapsto \mathcal{Y} \) from
input space \( \mathcal{X} \subseteq \mathbb{R}^d \) to output space \( \mathcal{Y} \) parametrized by a vector \( w \in \mathcal{W} \subseteq \mathbb{R}^d \). The
performance of \( w \) is assessed using a loss function \( \ell : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}_+ \). In empirical
risk minimization, we minimize the average loss, namely,

\[
\arg\min_{w \in \mathcal{W}} \mathcal{L}(w; \mathcal{D}) := \frac{1}{n} \sum_{(x_i, y_i) \in \mathcal{D}} \ell(w; (x_i, y_i)).
\]

As it was mentioned, for mitigating the effects of biases in the dataset on the learning
procedure we ought to include additional objectives such as recall rate for class imbalance
or some fairness violation measures for fairness-aware learning\(^3\). Thus, in this setting we
want to solve the following multiobjective optimization:

\[
w^* = \arg\min_{w \in \mathcal{W}} \mathbf{h}(w),
\]

\(^3\)Examples of fairness measures are provided in Chapters 5 and 6
where the objective vector is $\mathbf{h}(\mathbf{w}) = [h_1(\mathbf{w}), \ldots, h_m(\mathbf{w})] : \mathcal{W} \rightarrow \mathbb{R}^m$. In our setting for bias mitigation, the first objective is the main loss or empirical risk of the problem (i.e. $h_1(\mathbf{w}) = \mathcal{L}(\mathbf{w})$) and the rest of objectives are added for mitigating the effects of bias. For this chapter we focus on general objectives, and in the following chapters we design application-specific objective vectors depending on the problem we are working on.

The ultimate goal is to learn a classifier that has the optimal compromises between different objectives. Since no single solution would generally minimize all objectives simultaneously, we start by defining the notion of Pareto efficiency, that mathematically forms the optimal trade-off. Prior to that, we need the definition of dominance between two arbitrary solutions of a vector-valued functions (e.g., see [112]). For a single-objective optimization we can compare two solutions based on their scalar value objective functions. In a multiobjective setting, similarly, we can define the notion of dominance as follows:

**Definition 1 (Dominance).** Consider a vector-valued objective function $\mathbf{h} : \mathbb{R}^d \rightarrow \mathbb{R}^m$ with $m$ objectives $\mathbf{h}(\mathbf{w}) = [h_1(\mathbf{w}), \ldots, h_m(\mathbf{w})]$. We say the solution $\mathbf{w}_1$ dominates the solution $\mathbf{w}_2$ if $h_i(\mathbf{w}_1) \leq h_i(\mathbf{w}_2)$ for all $1 \leq i \leq m$, and $h_j(\mathbf{w}_1) < h_j(\mathbf{w}_2)$ for at least one $1 \leq j \leq m$. We denote this dominance as:

$$\mathbf{h}(\mathbf{w}_1) \prec_m \mathbf{h}(\mathbf{w}_2).$$  (4.3)

As depicted in Figure 4.1 as an example, in a trade-off between empirical loss of a learning problem as one objective and any fairness constraint as the second objective, we are seeking the Pareto efficient solutions that belongs to the Pareto frontier of the problem. This points on the red line cannot be dominated by other points in the feasible set. The scalarization of this multiobjective optimization such as constrained optimization can potentially end up in any non-efficient points in the feasible region; thereby lacking any guarantee on the optimally of trade-offs. Only considering one objective could end up in their respective optimal solutions $\mathbf{w}^*_1$ and $\mathbf{w}^*_2$. Thus, not only it is important to find a solution from Pareto frontier of the problem, but also we should be able to choose the desired level of compromises between objectives.

We now turn to defining the notion of Pareto efficiency in classification problems.

**Definition 2 (Pareto efficiency).** Consider any objective vector with $m$ objectives $\mathbf{h}(\mathbf{w}) = [h_1(\mathbf{w}), \ldots, h_m(\mathbf{w})]$ that ought to be minimized. The solution $\mathbf{w}^*$ is called Pareto efficient if no other feasible solution $\mathbf{w} \in \mathcal{W}$ can dominate $\mathbf{w}^*$. That is for this objective vector we have: $\exists \mathbf{w} \in \mathcal{W}$ such that $\mathbf{h}(\mathbf{w}) \prec_m \mathbf{h}(\mathbf{w}^*)$.  

36
is the optimum solution that minimizes the empirical risk, while \( w^*_2 \) is the optimum solution to satisfy the fairness constraint. Each solution on the Pareto frontier cannot be dominated in terms of both empirical risk and the fairness constraint by any other feasible solution.

In Section 4.2, we propose an iterative efficient algorithm that guarantees converges to a Pareto efficient solution of the described problem. But before that, we characterize the existence of such solution for a given problem with different objectives. In particular, the following theorem provides a necessary condition for existence of a Pareto efficient solution.

**Theorem 1.** Let \( h(w) = [h_1(w), \ldots, h_m(w)]^\top \) be the objective vector we ought to minimize. If the individual functions are bounded below, i.e. \( h_i(w) \geq C, \forall i \in [m], w \in \mathcal{W}, \) and \( C \in \mathbb{R}, \) then the set of Pareto efficient fair solutions is non-empty.

**Proof.** The proof is deferred to Section 4.6.1.

The curious readers is referred to the Section 4.6.1 for more details about implications of this theorem.

**4.2 Learning Pareto Efficient Classifiers**

Having multiple, rather contradictory objectives it seems hard to find an optimal solution for (4.2) that has the best compromises between all the objectives. As mentioned earlier,
most previous works reduce the problem into an instance of constrained optimization problem. In contrast, we take an alternative approach and introduce a bilevel optimization, by which we can find the solution to the vector optimization problem of (4.2), with a guarantee of convergence to a Pareto efficient point. This approach leads to a significantly efficient method and allows to find an optimal trade-off between different objectives. We then extend the proposed idea to construct Pareto frontier.

4.2.1 Bilevel Optimization

The proposed algorithm is motivated by key drawbacks of scalarization methods. In scalarization methods, as one of the simplest methods to tackle vector-valued optimization problems, one aims at reducing the optimization problem to a single objective one with combining the objectives into a single objective by assigning each objective function a nonnegative weight (e.g., convex combination of the objectives $\sum \alpha_i h_i(w), \alpha_i \geq 0, \sum \alpha_i = 1$). By optimizing over various values of the parameters used to combine the objectives, one is guaranteed to obtain solution from the entire Pareto front. While being conceptually appealing, however, it is difficult to decide the weights apriori or it requires investigating exponentially many parameters [55]. It is also observed evenly distributed set of weights fails to produce an even distribution of Pareto minimizers in the front [32]. Finally, it might be impossible to find the entire Pareto front if some of the objectives are nonconvex.

To tackle aforementioned issues, we propose a bilevel programming idea to adaptively learn the combination weights that corresponds to a single solution in Pareto front. The main idea stems from the fact that a solution is a first-order Pareto stationary, if the convex hull of the individual gradients contains the origin. Specifically, let $w^*$ to be a Pareto efficient solution of the optimization in (4.2). We know that there exists a vector $\alpha^* \in \mathbb{R}^m$, where $w^* = \arg\min_w \Psi(h(w), \alpha^*) \triangleq \sum_{i=1}^m \alpha^*_i h_i(w)$. Therefore, the key question is how to determine the optimal weights, $\alpha^* = [\alpha^*_1, \ldots, \alpha^*_m]^\top$. To this end, we first note that based on Karush-Kuhn-Tucker (KKT) conditions [20], the optimal weights should belong to the following set:

$$\left\{ \sum_{i=1}^m \alpha_i g_i(w^*) = 0, \ \alpha_i \geq 0, \ \forall \ 1 \leq i \leq m, \ \sum_{i=1}^m \alpha_i = 1 \right\},$$  \hspace{1cm} (4.4)

where $g_i(w^*) = \nabla h_i(w^*)$ is the normalized gradient vector of the $i$th objective at point $w^*$ in the objective vector $h(w)$.

The above condition is necessary condition for a point $w^*$ to be a Pareto efficient
point for the optimization problem (i.e., Pareto stationary). To find an optimal weight, following the condition in (4.4), we propose the bilevel optimization problem:

\[
\mathbf{w}^* \in \operatorname{arg \ min}_w \Psi (\mathbf{h}(\mathbf{w}), \alpha^*(\mathbf{w})) = \sum_{i \in [m]} \alpha_i^*(\mathbf{w}) h_i(\mathbf{w})
\]

\[
s.t. \quad \alpha^*(\mathbf{w}) \in \operatorname{arg \ min}_{\alpha \in \Delta_m} \Phi (\mathbf{h}(\mathbf{w}), \alpha) = \left\| \sum_{i=1}^{m} \alpha_i \mathbf{g}_i(\mathbf{w}) \right\|^2_2,
\]

(4.5)

where \( \Delta_m = \left\{ \mathbf{p} = [p_1, \ldots, p_m]^\top \in \mathbb{R}^m : p_i \geq 0, \sum_{i=1}^{m} p_i = 1 \right\} \) is the \( m \)-dimensional simplex.

Bilevel optimization consists of two nested optimization problems, namely inner and outer levels. Each level has its own objective function, but the solution of the inner level is being used in the optimization of the outer level. The following theorem shows that the solution of the outer-level optimization problem belongs to the Pareto efficient solution set of the optimization in (4.2).

**Theorem 2.** The solution returned by the optimization (4.5) is a Pareto efficient solution of the problem in (4.2).

**Proof.** Denoting the solution of optimization (4.5) by \( \mathbf{w}^*_b \) and the set of Pareto efficient fair solutions of vector optimization (4.2) by \( \Omega_P \). If \( \mathbf{w}^*_b \notin \Omega_P \), then there would be a \( \mathbf{w}_P \in \Omega_P \) that dominates \( \mathbf{w}^*_b \), which means that:

\[
h_i(\mathbf{w}_P) \leq h_i(\mathbf{w}^*_b), \quad \forall i \in [m]
\]

\[
h_i(\mathbf{w}_P) < h_i(\mathbf{w}^*_b), \quad \text{for at least one } i \in [m]
\]

Then, for any \( \alpha \in \Delta_m \), we can write:

\[
\sum_{i=1}^{m} \alpha_i h_i(\mathbf{w}_P) < \sum_{i=1}^{m} \alpha_i h_i(\mathbf{w}^*_b)
\]

\[
\Psi (\mathbf{h}(\mathbf{w}_P), \alpha) < \Psi (\mathbf{h}(\mathbf{w}^*_b), \alpha),
\]

(4.6)

where it contradicts the assumption that \( \mathbf{w}^*_b \) is the solution of optimization (4.5). Hence, \( \mathbf{w}^*_b \in \Omega_P \). \( \square \)

To solve the optimization in (4.5) using gradient descent algorithm, for every gradient step we take in the outer level, we have to find the optimal values for \( \alpha \) based on the optimization of the inner level. The following lemma shows that gradient of the outer
level objective computed at the optimal solution of inner level problem, is a descent direction to all objectives in \( h(w) \).

**Lemma 1** (Pareto Descent). Using the solution of the inner level, \( \alpha^* \), the gradient of the outer level is either zero or a descent direction for all the objectives of \( h(w) \). That is, for \( \nabla \Psi (h(w), \alpha^*) \), we have

\[
- \langle \nabla \Psi (h(w), \alpha^*), g_i(w) \rangle \leq 0, \quad \forall i \in \{1, \ldots, m\}.
\] (4.7)

**Proof.** First, for the sake of simplicity, we use the notation of \( g(t) = g(w(t)) \). We define the set of feasible solutions for the inner level optimization problem in (4.5):

\[
\Theta(w(t)) = \left\{ \sum_{i=1}^{m} \alpha_i g_i(t) \mid \alpha_i \geq 0 \quad \forall i \in \{1, \ldots, m\}, \sum_{i=1}^{m} \alpha_i = 1 \right\}
\] (4.8)

Now, we consider the solution of the inner level optimization of (4.5) to be \( \alpha^* (w(t)) \) for the iteration \( t \). Then the gradient of outer level can be written as:

\[
g(t) = \nabla_w \Psi \left( h \left( w(t) \right), \alpha^* \left( w(t) \right) \right) = \sum_{i=1}^{m} \alpha_i^* (w(t)) g_i \left( w(t) \right)
\] (4.9)

Hence, we consider that, if (4.7) was wrong, then there would be at least one \( g_i(t) \in \Theta(w(t)) \), where \( \langle \bar{g}(t), g_i(t) \rangle < 0 \). Then, we would consider the following optimization problem:

\[
\min_{\gamma \in [0,1]} \left\| (1 - \gamma)g_i(t) + \gamma \bar{g}(t) \right\|_2^2,
\] (4.10)

note that \( (1 - \gamma)g_i(t) + \gamma \bar{g}(t) \in \Theta(w(t)) \), thus, it is similar to the inner level optimization problem in (4.5). On the other side, based on (4.9), we know that \( \| \bar{g}(t) \|_2 \) is the solution to the optimization in the inner level of (4.5). It shows that the function in (4.10) is monotonically decreasing in \( \gamma \), and the minimum happens at \( \gamma^* = 1 \). By writing the first order condition on this point we have:

\[
2 \left( \bar{g}(t) - g_i(t) \right)^\top \left( g_i(t) + \gamma (\bar{g}(t) - g_i(t)) \right) \leq 0
\]
\[
\left( \bar{g}(t) \right)^\top \left( \bar{g}(t) - g_i(t) \right) \leq 0
\]
\[
\| \bar{g}(t) \|_2^2 \leq \left( \bar{g}(t) \right)^\top g_i(t),
\] (4.11)

where \( \oplus \) comes with using \( \gamma^* = 1 \). Inequality in (4.11) contradicts our assumption
Algorithm 2: Bilevel Pareto Descent Optimization (PDO)

\begin{align*}
\text{input } & \hat{\alpha}^{(0)} \in \mathbb{R}^m, w^{(0)} \in \mathbb{R}^d, \rho, \eta, K \\
\text{function } & \text{PDO}(h(w), \eta, \rho): \\
& \text{for } t = 0, 1, \ldots, T - 1 \text{ do} \\
& \quad \text{for } k = 0, 1, \ldots, K - 1 \text{ do} \\
& \quad \quad \alpha^{(k+1)} = \alpha^{(k)} - \rho \cdot \nabla \Phi \left( h \left( w^{(t)} \right), \alpha \right) \bigg|_{\alpha = \alpha^{(k)}} \\
& \quad \quad \alpha^{(k+1)} = \Pi_{\Delta_m} \left( \alpha^{(k+1)} \right) \\
& \quad \text{end} \\
& \quad \text{set } \hat{\alpha}^{(t)} = \alpha^{(K)} \\
& \quad w^{(t+1)} = w^{(t)} - \eta \cdot \nabla \Psi \left( h \left( w \right), \hat{\alpha}^{(t)} \right) \bigg|_{w = w^{(t)}} \\
& \text{end} \\
& \text{return } w^{(T)}
\end{align*}

on \((\bar{g}^{(t)})^\top g^{(t)}_i < 0\). Therefore, \(\bar{g}^{(t)}\) is a descent direction to all objectives, meaning, \(\langle \bar{g}^{(t)}, g^{(t)}_i \rangle < 0\) for every \(1 \leq i \leq m\).

Remark 1. Lemma 1 implies that in every step the gradient of the outer level is either zero, which means we reach a Pareto stationary point, or a descent direction to all objectives, which can be used to decrease all the objective by moving in the negative direction descent direction.

Equipped with this descent direction, we can guarantee that all the objectives at every iteration are non-increasing, until we reach to a point that this descent direction is 0, which means that we cannot improve all objectives, without hurting others. This indicates that we are in a Pareto stationary point of the problem. Note that from the first-order optimality condition in (4.4), there exists a pair of \((\alpha^*, w^*)\) such that \(\alpha^* \in \Delta_m\) and \(\sum_{i=1}^{m} \alpha^*_i g_i(w^*) = 0\).

In order to solve the optimization in (4.5), we propose an approximate procedure in Algorithm 2. Having \(m < d\) in our optimization, the inner level would be a strongly convex function, and hence, we can converge to its global minimum quickly. In fact, using gradient descent, we can converge to an \(\epsilon\)-accurate solution in \(O \left( \log \frac{1}{\epsilon} \right)\) steps [21]. This error will then be propagated to the outer level using \(\hat{\alpha}\). We will bound this error in the Section 4.4 alongside the convergence analysis of the algorithm.
4.3 Pareto Frontier

Using Algorithm 2 (PDO), we can converge to a single Pareto stationary point. However, from a practical point of view, it might be required to calculate different solutions from the entire spectrum of the Pareto frontier to be able to pick a solution with a desired trade-off. For instance, in the fairness-aware learning problem, in some cases, the goal might be to keep the model as accurate as possible while imposing some degree of fairness constraints; in some others, we want to strictly satisfy the fairness constraints, even if it hurts the accuracy of the model; in yet some others, we want a half-point compromises in between of these two extreme points. Thus, it is important to find a set of Pareto efficient points, which is called Pareto frontier. Having access to the Pareto frontier would help a decision maker to choose from a variety optimal solutions with different compromises between objectives. Despite its importance, extracting points on the Pareto frontier is the one of main challenges of multiobjective optimization. In this section, we first explore the geometry of the Pareto frontier and the relationship between optimal weights of objectives on a Pareto efficient solution (α (w∗)) and the Pareto frontier surface. Then, we propose a novel algorithm that can extract points from the Pareto frontier of different objectives with only access to the first-order information.

4.3.1 Pareto Frontier Geometry

In this section, we investigate the geometrical characteristics of the Pareto frontier curve. For a simple case of bi-objective optimization, where the Pareto frontier curve of these two objective is smooth and convex we can have a following property:

**Proposition 1.** For a smooth and convex Pareto frontier of a vector objective, an ordering in the optimal weights, α, entails an ordering over their corresponding objectives.

**Proof.** To show this property on smooth and convex Pareto curves, we consider a simple case of two objectives, however, it can be easily generalized to more than two objectives. For this case, we want to find a point from the Pareto frontier. To do so, as discussed before, we minimize the scalarized version of the objectives using mixing parameters 1 − α and α for h₁ (w) and h₂ (w), respectively. As it was used by other studies [100, 31], we can replace α by \( \frac{\sin \theta}{\sin \theta + \cos \theta} \), \( \theta \in \left[ 0, \frac{\pi}{2} \right] \), and optimize for \( \theta \) instead of \( \alpha \). Then the scalar
Figure 4.2: Schematic view of relationship between weighting parameters and Pareto optimal solution properties. If we replace $\alpha$ in a bi-objective optimization problem with $\sin \theta + \cos \theta$, it can be shown that for every $\theta \in \left[0, \frac{\pi}{2}\right]$ the Pareto efficient solution $w_\theta$ is the tangent point of the Pareto frontier curve and the line perpendicular to the rotated coordinate of $h_1(w)$ with angle $\theta$, i.e., $h_\theta(w) = \cos \theta \cdot h_1(w) + \sin \theta \cdot h_2(w)$.

objective we are trying to minimize for two objectives, $h_1(w)$ and $h_2(w)$, becomes:

$$h(w) = \frac{\cos \theta \cdot h_1(w) + \sin \theta \cdot h_2(w)}{\sin \theta + \cos \theta}, \quad \forall \theta \in \left[0, \frac{\pi}{2}\right], w \in \mathcal{W}. \quad (4.12)$$

Now, it is clear that the transformation from $h_1(w)$ and $h_2(w)$ to the new objective is a rotation by the angle of $\theta$ following by a normalization by $\sin \theta + \cos \theta$. As it can be inferred from the Figure 4.2, it seems that the solution of the optimization is the minimum point on the curve based on the new rotated coordinate.

For two extreme points in $\theta$, that are, $\theta_1 = 0$ and $\theta_2 = \frac{\pi}{2}$, the minimization objective reduces to $h(w) = h_1(w)$ and $h(w) = h_2(w)$, respectively. Hence, for these two cases, we find the minimum solution for each objective, respectively. For values between these two extremes, $\theta \in \left(0, \frac{\pi}{2}\right)$, the minimum point is the tangent point of the curve with the line perpendicular to the rotated coordinate, as it is shown in the Figure 4.2. To show that this point is the Pareto solution based on the current $\alpha$ value, we can write down the slope of the curve on this tangent point:

$$\frac{\partial h_2(w)}{\partial h_1(w)} \bigg|_{w=w_\theta} = \tan \left(\theta + \frac{\pi}{2}\right)$$
\[
\cot \theta = -\frac{1 - \alpha}{\alpha}, \quad (4.13)
\]

then, replacing the partial derivatives with gradients of each function with respect to \( \mathbf{w} \), we can get:

\[
\nabla h_2 (\mathbf{w}) = -\frac{1 - \alpha}{\alpha} \cdot \nabla h_1 (\mathbf{w}) \\
(1 - \alpha) \nabla h_1 (\mathbf{w}) + \alpha \nabla h_2 (\mathbf{w}) = 0, \quad (4.14)
\]

which is the condition derived from KKT conditions in (4.4) for the Pareto optimality of a point. Hence, it is clear that in the case of smooth and convex Pareto curve, for each \( \alpha \) and subsequently for each \( \theta \), the Pareto efficient point is the tangent point described above. For the two extreme points mentioned before we have:

\[
\theta_1 = 0 \iff \alpha_1 = 1 \iff \frac{\partial h_2 (\mathbf{w}_1)}{\partial h_1 (\mathbf{w}_1)} = -\infty \\
\theta_2 = \frac{\pi}{2} \iff \alpha_2 = 0 \iff \frac{\partial h_2 (\mathbf{w}_2)}{\partial h_1 (\mathbf{w}_2)} = 0, \quad (4.15) \quad (4.16)
\]

then for every angle in between, \( \theta_1 \leq \theta \leq \theta_2 \), we have \(-\infty \leq \frac{\partial h_2 (\mathbf{w}_\theta)}{\partial h_1 (\mathbf{w}_\theta)} \leq 0\), which implies the non-increasing curve. Next, for two angles in this range, \( \theta_1 \leq \theta_3 \leq \theta_4 \leq \theta_2 \), we will have \( 1 \geq \alpha_3 \geq \alpha_4 \geq 0 \). Then, it can be easily verified that we have \( h_1 (\mathbf{w}_{\theta_4}) \geq h_1 (\mathbf{w}_{\theta_3}) \) and \( h_2 (\mathbf{w}_{\theta_4}) \leq h_2 (\mathbf{w}_{\theta_3}) \), which entails the ordering on the objectives as well.

\[ \square \]

Although, we can show this property for a convex and smooth Pareto frontier, most of the time, these conditions are not met. Hence, the aforementioned mapping might not be valid in general, nonetheless, it can shed light on how different weight parameters are mapped to different regions in the Pareto frontier curve.

### 4.3.2 Pareto Frontier Extraction

The goal of finding different points from the Pareto frontier and constructing a Pareto frontier set for a vector minimization problem has been investigated in vast number of studies so far [66]. However, achieving this goal is still a challenging problem that requires an algorithmic approach with tractable computational complexity for different scenarios. In general, we can categorize the main approaches that aim to find points
from the Pareto frontier or trace the points on this set, into four groups. There are quite number of heuristic approaches as well, however, for this paper we focus on the main analytic ones.

**Normal Boundary Intersection (NBI):** This approach has been proposed by [32] to find evenly distributed points on the Pareto frontier and address the problem of different scales in the objectives by previous methods. In this method, we need to adjust each dimension of the objective vector with respective global minimum of that objective, hence their minimum values translate to zero. Next, they need to find points on the convex hull of individual minima from different objectives. Using the points and the normal vector of this convex hull toward the origin, they solve an optimization problem to move from those points along the normal vector until they reach to a point from the feasible set boundary. The resulting solutions are stationary points of the objective vector. The main drawbacks of this approach is that finding the global minimum of each objective is computationally expensive. More importantly, finding the solutions in the parameter space corresponding to the points from the mentioned convex hull in the objective space is not trivial.

**Geometrical Exploration:** There are several different geometrical strategies to find points from the Pareto frontier set, given we already found one. These methods rely on the geometrical properties such as rank condition on the Hessian matrix of the Pareto frontier and use it to traverse it, starting from a point in that set. In [66], the author elaborate on these properties in detail and propose two different Homotopy strategies. One of them is the local exploration of the Pareto frontier using a linear program based on Jacobian and Hessian matrices of the objectives. In [107], authors used the exact same idea and propose an algorithm to solve this linear program, in order to trace points on the Pareto frontier. Using second-order information of the stationary points, especially in high-dimensional problems with high number of objectives is not computationally feasible.

**Weight Perturbation:** The second geometrical idea from [66] is weight perturbation. As it was mentioned in the previous section, there is an ordering relationship between mixing weights $\alpha_i^*$, $i \in [m]$ of the final solution and the location of that solution on the Pareto frontier. In [66], the author use the same idea and propose an algorithm to find nearby points on Pareto frontier by perturbing the mixing weight of the stationary point
found by other algorithms like our Algorithm 2. Similar to the previous approach, the second-order information needed for this algorithm to work is disadvantageous.

**Preference-based solutions:** A novel class of approaches to find points on the Pareto frontier on specified part of the objective space are called preference-based methods. These approaches use a predefined preference vector over different objectives. The value of each objective’s preference with respect to other objectives’ shows how much we want to be close to the minimum value of that objective compared to other objectives’ in the Pareto frontier. In [100], authors use different preference vectors to segment the Pareto frontier into equal parts, and then, design an optimization problem to enforce the final solution of each run to fall in one of those desired parts. By repeating this algorithm they can find evenly distributed points from different parts of the Pareto frontier. However their approach heavily depends on the location of the initial solution. Recently, in [108], authors use the preference vector and design an optimization to converge to a single specific point from the Pareto frontier.

Our approach to find points from the Pareto frontier belongs to the preference-based methods similar to [108], but instead of targeting a single point we use this approach to trace the points on the Pareto frontier before reaching that specific point. Traversing the points on the Pareto frontier in our approach is similar to the geometrical exploration approaches. Despite this similarity, our approach only uses *first-order* information, which is computationally efficient than those methods.

### 4.3.2.1 Preference-Based Pareto Descent Optimization

We are pursuing to find points from the Pareto frontier using a user predefined preference over different objectives. These preferences can be represented as a vector \( \pi \in \mathbb{R}_+^m \). In this setting, the ultimate goal is to reach to a point from the Pareto frontier, where the ratio between objectives’ value is proportional to the ratio of corresponding preference values in this vector. That is for this point we aim to satisfy:

\[
\pi_1 h_1 (w_{PB}^*) = \pi_2 h_2 (w_{PB}^*) = \ldots = \pi_m h_m (w_{PB}^*),
\]

(4.17)

where \( w_{PB}^* \) is a preference-based solution from the Pareto frontier that satisfies this condition\(^4\). This condition implies that if we increase the weight for one objective, we

\(^4\)The assumption is that such a point on the Pareto frontier that satisfies this condition exists. If this point does not exist, in practice, we reach to its closest point on the Pareto frontier.
Figure 4.3: Example objectives in the parameter space. The red line is the projection of Pareto frontier on the parameter space.

require to find a Pareto optimal solution with lower objective value for that specific objective. In addition, the condition in (4.17) suggests that this point is the intersection of the Pareto frontier and a line in the objective space, determined by:

$$h = [h_1, \ldots, h_m] = p \circ (c1_m),$$  

(4.18)

where $p = [1/\pi_1, \ldots, 1/\pi_m]$, $1_m$ is an all ones vector with size $m$, $c \in \mathbb{R}$ is an independent variable, and $a \circ b$ shows the Hadamard or element-wise multiplication of two vectors $a$ and $b$. We call this line the “preference line”. Using the following simple example for a bi-objective problem ($m = 2$), this line and its intersection point for an arbitrary $\pi$ and $c$ is depicted in Figure 4.4. This line intercepts the origin and $p$.

**Example 1.** For the sake of exposition for generating Pareto frontier, we use a simple and classic example [125], where we have a bi-objective optimization with two objectives as:

$$h_1(w) = 1 - e^{-\frac{\|w - \nu\|^2}{s^2}}, \quad h_2(w) = 1 - e^{-\frac{\|w + \nu\|^2}{s^2}},$$  

(4.19)

where $\nu \in \mathbb{R}^d$ and $s \in \mathbb{R}$ are the mean and variance parameters. For instance, the objective values for $w \in \mathbb{R}^2$, $\nu = 1_2$, and $s = 1.5$ are depicted in Figure 4.3.

In order to converge to the desired point $w_{PB}^\ast$, we should define a new objective that measures how the distribution of objective values on each solution point are deviating from the condition in (4.17). We can project the vector $\pi \circ h$ into a $\Delta_m$ simplex, for
Each arbitrary solution point \( \mathbf{w} \), using a Softmax function:

\[
\sigma_i(\mathbf{w}, \pi) = \frac{e^{\pi_i h_i(\mathbf{w})}}{\sum_{j \in [m]} e^{\pi_j h_j(\mathbf{w})}}, \quad \forall i \in [m],
\]

then the \( \sigma_i(\mathbf{w}) \) values can be considered as a probability distribution. Now, the condition in (4.17) reduces to:

\[
\sigma_i(\mathbf{w}_{\text{PB}}, \pi) = \frac{1}{m}, \quad \forall i \in [m],
\]

which is the uniform distribution. Thus, the best choice for the objective function to measure the discrepancy between probability values in (4.20) and (4.21) seems to be KL-divergence between these two distributions:

\[
h_{\text{KL}}(\mathbf{w}, \pi) = \text{KL}(\sigma(\mathbf{w}, \pi) \| \sigma(\mathbf{w}_{\text{PB}}, \pi))
\]

\[
= \sum_{i \in [m]} \sigma_i(\mathbf{w}, \pi) \log \left( \frac{\sigma_i(\mathbf{w}, \pi)}{\sigma_i(\mathbf{w}_{\text{PB}}, \pi)} \right)
\]

\[
= \sum_{i \in [m]} \sigma_i(\mathbf{w}, \pi) \log (m \sigma_i(\mathbf{w}, \pi)). \quad (4.22)
\]
We want to minimize this objective, which indicates that we want to minimize the entropy of $\mathbf{\sigma}(\mathbf{w}, \mathbf{\pi})$ and maximize the cross entropy between $\mathbf{\sigma}(\mathbf{w}, \mathbf{\pi})$ and $\mathbf{\sigma}(\mathbf{w}_{PB}, \mathbf{\pi})$. This objective has its minimum value of zero on all the points on the line defined in (4.18).

The way to minimize the objective $h_{KL}$ in (4.22), in addition to the main objective vector $\mathbf{h}$, is to add it as another objective to the objective vector and have an $m + 1$-dimensional vector to minimize. Using this approach we make sure that at each step we find a direction that is descent for all the objectives in $\mathbf{h}$ as well as $h_{KL}$. Hence, in general we consider the following objective vector as the main vector to minimize for our problem:

$$h_{PB}(\mathbf{w}, \mathbf{\pi}) = [h_1(\mathbf{w}), \ldots, h_m(\mathbf{w}), h_{KL}(\mathbf{w}, \mathbf{\pi})] \in \mathbb{R}^{m+1},$$

(4.23)

where $PB$ in $h_{PB}$ stands for a preference-based objective with the preference vector indicated by $\mathbf{\pi}$. To minimize this vector, we need to compute the gradient of the $h_{KL}$ at each iteration, similar to other objectives indicated in (4.5). The following proposition compute this gradient for this problem.

**Proposition 2.** The gradient vector of the objective $h_{KL}(\mathbf{w}, \mathbf{\pi})$ with respect to any arbitrary solution point $\mathbf{w} \in \mathbb{R}^d$, is denoted by $g_{KL}(\mathbf{w}, \mathbf{\pi}) \in \mathbb{R}^d$ has the following form:

$$g_{KL}(\mathbf{w}, \mathbf{\pi}) = \nabla_{\mathbf{w}} h_{KL}(\mathbf{w}, \mathbf{\pi}) = \sum_{i \in [m]} \lambda_i g_i$$

s.t. $\lambda_i = \pi_i \sigma_i(\mathbf{w}, \mathbf{\pi}) (\log (m \sigma_i(\mathbf{w}, \mathbf{\pi})) - h_{KL}(\mathbf{w}, \mathbf{\pi}))$,

where $g_i, i \in [m]$ are the gradients of objectives in the objective vector $\mathbf{h}$.

**Proof.** We start by the gradient of the Softmax function. For an arbitrary input vector $\mathbf{z} = [z_1, \ldots, z_m]$, applying the Softmax function would yield to $\sigma_i \triangleq \sigma_i(\mathbf{z}) = e^{z_i} / \sum_{j \in [m]} e^{z_j}, \ \forall i \in [m]$. Then, it is straightforward to show the derivative of this function with respect to each input is:

$$\frac{\partial \sigma_i}{\partial z_j} = \begin{cases} \sigma_i (1 - \sigma_i), & \text{if } i = j. \\ \sigma_i \sigma_j, & \text{otherwise}. \end{cases}$$

(4.25)

Now, if we consider $z_i = \pi_i h_i(\mathbf{w}), \ \forall i \in [m]$, we can write the partial gradient of the objective $h_{KL}$ with respect to $z_j$ is:

$$\frac{\partial h_{KL}(\mathbf{w}, \mathbf{\pi})}{\partial z_j} = \sum_{i \in [m]} \frac{\partial \sigma_i}{\partial z_j} \cdot (\log (m \sigma_i) + 1)$$
\[ \sigma_j (1 - \sigma_j) (\log (m\sigma_j) + 1) - \sum_{i \in [m], i \neq j} \sigma_i \sigma_j (\log (m\sigma_i) + 1) \]
\[ = \sigma_j (\log (m\sigma_j) + 1) - \sum_{i \in [m]} \sigma_i \sigma_j (\log (m\sigma_i) + 1) \]
\[ = \sigma_j (\log (m\sigma_j) + 1) - \sigma_j \sum_{i \in [m]} \sigma_i (\log (m\sigma_i) + 1) \]
\[ = \sigma_j (\log (m\sigma_j) + 1) - \sigma_j (h_{KL} (w, \pi) + 1) \]
\[ = \sigma_j (\log (m\sigma_j) - h_{KL} (w, \pi)), \quad (4.26) \]

where in ①, we use (4.25), and in ②, we use (4.22) and the fact that \( \sum_{i \in [m]} \sigma_i = 1 \).

Now, by considering the definition of \( z_j \) and noting that \( g_i = \partial h_i (w) / \partial w \), the proof is complete. \( \square \)

The form of the descent gradient direction for \( h_{KL} \) indicated by Proposition 2 shows that similar to the descent direction of the main objective vector \( h \), this gradient is a linear combination of objective’s gradients. We will use this gradient direction in our algorithm to minimize the loss for preference-based objective.

In general, minimizing the preference-based objective vector \( h_{PB} \) would converge to the desired point \( w^*_{PB} \). However, there are two cases depending on the position of the initial point in the problem that might not be able to converge to that desired solution. These two cases happen when (I) the initial solution is too close to the preference line defined in (4.18); or (II) it is far away from the desired solution \( w^*_{PB} \). The reason that minimizing the vector \( h_{PB} \) might not converge in these two cases is that we reach to a local minimizer of either of its two objectives (a point from the preference line in (4.18) for \( h_{KL} \) or a point from the Pareto frontier of \( h \)) before the desired point \( w^*_{PB} \). In either of these cases, the resulting descent direction would be zero and it cannot escape that point. These two situations are depicted in Figure 4.5(a), where the model converges to either \( w_I \) or \( w_{II} \) for cases I and II, respectively. Hence, we need to develop an algorithm to address such cases. It is worth mentioning that, even in these two cases the algorithm is converging to a stationary point of the preference-based objective \( h_{PB} \), which is in line with the theoretical results. However, since the goal is to reach to a desired point on the Pareto frontier of \( h \), these stationary points are not satisfying that goal.

We design Algorithm 3 that introduces Preference-Based Pareto Descent Optimization (PB-PDO). In this algorithm, the user determines a preference vector \( \pi \) over set of different objectives we have, and the algorithm finds the best \( w^*_{PB} \) on the Pareto frontier of \( h \) that satisfies the condition in (4.17). In order to avoid the two cases described before...
Figure 4.5: Using objectives in Example 1 with $w \in \mathbb{R}^{20}$ to (a) demonstrate two challenging cases, which cannot converge to the desired point $w_{PB}^*$ by only using $h_{PB}$ as the objective; and (b) how PB-PDO by adaptively choosing objectives can achieve this goal. In (a) we converge to points $w_I$ and $w_{II}$ for cases I and II, respectively, which are stationary points of the $h_{PB}$ but they are not the desired point.

we need to adaptively define our set of objectives for finding the descent direction at each iteration. In general, our algorithm at each iteration finds a descent direction $d = \nabla \Psi (h_{PB}(w, \pi), \alpha^*(w))$, where $\alpha^*(w)$ is determined similar to (4.5), for the objective vector $h_{PB}$. If either of the following conditions met, which are indicators of the aforementioned cases, we will tune our objective vector and accordingly its descent direction, to escape those points toward the desired point:

(I) The first case happens when we reach a point on the preference line in (4.18), other than the desired point on the Pareto frontier $w_{PB}^*$. The reason is that the initial point is close to the preference line (see Figure 4.5(a) and the trajectory from $w_0$ toward $w_1$). For this point we have $g_{KL} = 0$, however this point is not on the Pareto Frontier of the main objective vector $h$. It is worth noting that this point is a stationary point of $h_{PB}$, however, it is not a stationary point of $h$. Thus, whenever we have:

$$\|g_{KL}(w, \pi)\| \leq \epsilon_1,$$  \hspace{1cm} (4.27)

we use the main objective vector $h$ instead of the preference-based one $h_{PB}$, to find the descent direction. This would help us to reach to a point from the Pareto frontier of $h$ before reaching the preference line. In this condition $\epsilon_1$ is a small
Algorithm 3: Preference-Based Pareto Descent Optimization (PB-PDO)

input $\hat{\alpha}^{(0)} \in \mathbb{R}^m$, $w^{(0)} \in \mathbb{R}^d$, $\rho$, $\eta$, $T$, $\pi$, $\epsilon_1$, $\epsilon_2$

function PB-PDO($h(w)$, $\pi$, $\eta$, $\rho$, $\epsilon_1$, $\epsilon_2$):

for $t = 0, 1, \ldots, T - 1$ do
    Compute $g_{KL}(w^{(t)}, \pi)$ using (4.24)
    if $\|g_{KL}(w, \pi)\| \leq \epsilon_1$ then  // Case (I)
        $d^{(t)} = \text{DescentDirection}(h, w^{(t)}, \rho)$
    else
        $d^{(t)} = \text{DescentDirection}(h_{PB}, w^{(t)}, \rho)$
    end
    if $\|d^{(t)}\|/\|g_{KL}(w, \pi)\| \leq \epsilon_2$ then  // Case (II)
        $d^{(t)} = g_{KL}(w^{(t)}, \pi)$
    end
    $w^{(t+1)} = w^{(t)} - \eta \cdot d^{(t)}$
end
return $w^{(T)}$

function DescentDirection($f$, $w^{(t)}$, $\rho$):

for $k = 0, 1, \ldots, K - 1$ do
    $\alpha^{(k+1)} = \alpha^{(k)} - \rho \cdot \nabla_{\alpha} \Phi (f(w^{(t)}), \alpha)\bigg|_{\alpha = \alpha^{(k)}}$
    $\alpha^{(k+1)} = \Pi_{\Delta_m} (\alpha^{(k+1)})$
end
set $\hat{\alpha}^{(t)} = \alpha^{(K)}$
return $\nabla \psi (f(w^{(t)}), \hat{\alpha}^{(t)})$

arbitrary threshold. See Figure 4.5(b) and the trajectory from $w_0$ toward $w_{\pi}^{*PB}$, where using this condition would not allow the model to reach a point on the preference line before touching the Pareto frontier of $h$.

(II) The second case happens when we reach to a point from the Pareto frontier, before the desired point $w_{\pi}^{*PB}$ that satisfies the condition in (4.17). This scenario occurs when the initial solution is far away from the desired point, and hence, using descent directions would get us to a point from the Pareto frontier sooner than that desired point (see Figure 4.5(a) and the trajectory from $w_1$ toward $w_{II}$). In this case, when we reach to a point from the Pareto frontier, the $\ell_2$-norm of the descent direction would be almost zero since the point is stationary point of the main objective vector $h$. However, since the $h_{KL}$ is not minimized yet, the $\ell_2$-norm
of its gradient $g_{KL}$ is large. Therefore, the case happens when we have:

$$\frac{\|d\|}{\|g_{KL}(w, \pi)\|} \leq \epsilon_2,$$

(4.28)

where $\epsilon_2$ is a small arbitrary threshold. Whenever the condition in (4.28) is met, which means that we reach to such a point, instead of using the descent direction from one of the objective vectors ($h$, or $h_{PB}$), we only use $g_{KL}(w, \pi)$. This means that we are only minimizing $h_{KL}$. This case, in fact, is beneficiary for tracing the Pareto frontier, which we elaborate on later. In Figure 4.5(b), the trajectory from $w_1$ toward $w_{PB}^*$ shows that using this approach we can trace points from the Pareto frontier of the main objective $h$.

The full procedure is defined in Algorithm 3. Now, the following remarks regarding the novelties of the proposed algorithm (PB-PDO) over previous methods and how to trace points from the Pareto frontier are in order.

**Remark 2.** The proposed algorithm (PB-PDO), despite some of the previous approaches, does not require the minimizers of individual objectives. Also, since the preference vector is not bounded to a simplex, the ratio between preference values of different objectives could be any arbitrary positive number. Hence, this approach can be applied to objectives with different scales, which was a limitation for some other approaches. Finally, this algorithm only uses first-order information to reach to the desired point and traverse the Pareto frontier, which is a huge computational advantage over similar counterparts.

**Remark 3.** Using the proposed algorithm (PB-PDO), we can traverse points on the Pareto frontier, which is missing from other preference-based approaches such as [100, 108]. In fact, the closest approach to ours [108] uses ascent directions in some iterations to reach to the desired point, and hence, needs to set some constraints to avoid divergence. Their approach makes sure that does not touch the Pareto frontier before reaching to that desired point. However, our approach, using only descent directions, not only converges to the desired point, but also able to trace points on the Pareto frontier. The comparison between these two algorithms as well as Algorithm 2 (PDO) is depicted in Figure 4.6.

To find points from different parts of the Pareto frontier using the proposed algorithm we can set different preference vectors and run the algorithm several times. Then, by choosing non-dominating points in the trajectories of these runs, we can extract points from different parts in the Pareto frontier. In Chapter 5 the results of applying this
4.4 Convergence Analysis

We now turn to analyzing the convergence of the proposed algorithm PDO. In order to prove the convergence of the bilevel optimization introduced in (4.5), we first need to discuss the effect of the approximation of the inner level’s solution on that of the outer level. Because we are solving both levels with gradient descent, a residual error from the inner level will be propagated through the outer level updates. Before diving into the convergence analysis, we indicate the assumptions used in the analysis. The first set of assumptions is related to the outer-level objective function, which is the weighted sum of objective functions in vector optimization of (4.2).

Assumption 1.a. Objective functions \( h_i (w), i \in [m] \), are bounded by a constant \( D_h = \max_{w \in W} \| h(w) \| \).

Assumption 1.b. Objective functions \( h_i (w), i \in [m] \), are differentiable and have bounded gradients of \( \| g_i(w) \| \leq B_h, \forall i \in [m] \).
Assumption 1.c. Objective functions $h_i(w), i \in [m], are smooth with Lipschitz constant of $L_i, i \in [m]$. That is, for every $w_1, w_2 \in \mathcal{W}$ and for every $i \in [m]$, we have:

$$\|\nabla h_i(w_1) - \nabla h_i(w_2)\| \leq L_i \|w_1 - w_2\|.$$ (4.29)

Now, we elaborate on the set of assumptions for the inner level objective function $\Phi(h(w), \alpha)$ as follows:

Assumption 2.a. By having $m < d$, the inner-level objective function $\Phi(h(w), \alpha)$ is strongly convex by parameter $\mu_{\Phi}$. That is, $\nabla^2_{ww} \Phi(h(w), \alpha) \succeq \mu_{\Phi} I$.

Assumption 2.b. The inner objective function $\Phi(h(w), \alpha^*(w))$ is smooth with Lipschitz parameter $L_{\Phi}$. That is, for every $w_1, w_2 \in \mathcal{W}$, we have:

$$\|\nabla \Phi(h(w_1), \alpha^*(w_1)) - \nabla \Phi(h(w_2), \alpha^*(w_2))\| \leq L_{\Phi} \|w_1 - w_2\|.$$ (4.30)

Assumption 2.c. The second-order derivative of the inner-level objective function $\nabla^2_{w\alpha} \Phi(h(w), \alpha)$ is $L_{w\alpha}$-Lipschitz continuous and bounded by $\|\nabla^2_{w\alpha} \Phi(h(w), \alpha)\| \leq H_{\Phi}$.

The last assumption for the inner-level objective is being commonly used by other machine learning and optimization frameworks in [117, 75, 129, 56]. Now, using these assumptions, we turn into the convergence analysis of the Algorithm 2. First, we should bound the error on the solution of the inner level.

**Lemma 2.** Let $\Phi(h(w), \alpha)$ be a strongly convex and smooth function with respect to $\alpha$ with $\mu_{\Phi}$ and $L_{\Phi}$ as its strong convexity and smoothness parameters, respectively. Then, if the inner learning rate is $\rho \leq \frac{2}{L_{\Phi}}$, the solution found by inner level of Algorithm 2 after $K$ updates is bounded by:

$$\|\hat{\alpha}(t) - \alpha^*(w^{(t)})\|^2 \leq \exp\left(-\frac{K}{\kappa}\right) \|\alpha^{(0)} - \alpha^*(w^{(t)})\|^2,$$ (4.31)

where $\alpha^*(w^{(k)})$ is the optimal weights for point $w^{(k)}$ and $\kappa = \frac{\mu_{\Phi}}{L_{\Phi}}$ is the condition number of function $\Phi(h(w), \alpha)$.

**Proof.** This is the standard convergence rate for a strong convex and smooth function using gradient descent. For the detailed proof refer to Bubeck et al. [21].

Then, we can write the gradient of the outer level function based on the inner level using chain rule as follows:

$$\nabla \Psi(h(w), \hat{\alpha}(w)) = \nabla_w \Psi(h(w), \hat{\alpha}(w)) + \nabla \hat{\alpha}(w) \nabla_{\alpha} \Psi(h(w), \hat{\alpha}(w))$$ (4.32)
The issue here is that we do not have the exact value of $\nabla \hat{\alpha}(w)$, and that’s where the approximation error comes into effect. From the optimality of the inner level, which is $\nabla_\alpha \Phi(h(w), \alpha^*(w)) = 0$, if we take the gradient from both sides, we will have:

$$\nabla \alpha^*(w) = -\nabla^2_{ww} \Phi(h(w), \alpha^*(w)) \left[ \nabla^2_{\alpha\alpha} \Phi(h(w), \alpha^*(w)) \right]^{-1}. \quad (4.33)$$

This is valid for the optimal point only, but we can use this approximation on the solution point found by Algorithm 2. Then, equipped with Lemma 2, we can bound the error in this approximation for the gradient of the outer level using the following lemma.

**Lemma 3** (Gradient error). Consider that the inner- and outer-level functions in optimization (4.5) are following the Assumptions 1.a, 1.b, 2.a, and 2.c and that the solution of the inner level is found by the Algorithm 2, then the error in the gradient of the outer-level function is bounded by

$$\|\nabla \Psi(h(w), \alpha(w)) - \nabla \Psi(h(w), \alpha^*(w))\| \leq A_\Psi \|\alpha - \alpha^*\| \quad (4.34)$$

where $A_\Psi = B_h \sqrt{m} + \frac{D_h \sqrt{m}}{\mu_\Phi} L_{w\alpha}$.

**Proof.** The proof is provided in Section 4.6.2, where we use the equation and its approximation for non-optimal points in (4.33), in addition to Assumptions mentioned above.

This lemma shows that the error in the gradient of the outer level because of inexact solution of the inner level can be bounded by the error in the solution of the inner level, which is then bounded by (4.31).

In the assumptions, we did not talk about the smoothness of the outer-level function. Next, using the smoothness of each objective function separately, we show that the outer-level function is also smooth, which is deliberated by the following lemma.

**Lemma 4.** The outer level function $\Psi(h(w), \alpha)$ is smooth with Lipschitz constant $L_\Psi$, that is:

$$\|\nabla \Psi(h(w_1), \alpha^*(w_1)) - \nabla \Psi(h(w_2), \alpha^*(w_2))\| \leq L_\Psi \|w_1 - w_2\|, \quad (4.35)$$

where $L_\Psi = L_{max} + \frac{L_{\Phi} B_h \sqrt{m}}{\mu_\Phi}$, in which $L_{max} = \max \{L_1, \ldots, L_m\}$ and $L_i, i \in [m]$ is the smoothness parameter of the $i$-th objective from Assumption 1.c.

**Proof.** The proof is deferred to Section 4.6.3. \qed
Now, equipped with Lemmas 2, 3, and 4, we can turn into the convergence analysis of the optimization in Algorithm 2. Note that, we do not have any convexity assumption for the outer-level function in our assumptions set. Hence, we consider two cases, where the outer-level function is convex or non-convex. First, we consider the case where all the objectives are convex functions. Because in $\Psi(w, \alpha)$ the objectives are summed with positive weights from a simplex, $\Psi(w, \alpha)$ is convex as well. Hence, we have the following theorem for the convergence of the Algorithm 2.

**Theorem 3** (Convex). Let $h(w) = [h_1(w), \ldots, h_m(w)]$ be the convex objective vector that follows the properties in Assumptions 1.a, 1.b, and 1.c. Then the function $\Psi(h(w), \alpha)$ is a convex and smooth function with Lipschitz constant of $L_\Psi$ from Lemma 4. Also, the inner function of optimization (4.5) has the properties in Assumptions 2.a, 2.b, and 2.c. Then, for the sequence of solutions $w^{(0)}, \ldots, w^{(T)}$, generated by the Algorithm 2, by setting $\eta \leq 1/L_\Psi$, we have

$$
\Psi(h(\bar{w}), \bar{\alpha}(\bar{w})) - \Psi(h(w^*), \alpha^*(w^*))
\leq \frac{1}{2\eta T} \|w^{(0)} - w^*\|^2
+ \frac{RA_\Psi}{T} \exp \left( - \frac{K}{2\kappa} \sum_{t=0}^{T-1} \|\alpha^{(0)}(w^{(t)}) - \alpha^*(w^{(t)})\| \right),
$$

where $R$ is the bound on the domain of the solutions and $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$. Also, $\bar{\alpha} = [\bar{\alpha}_1, \ldots, \bar{\alpha}_m]$, and $\bar{\alpha}_i = \min_t \alpha_i^*(w^{(t)})$ for $1 \leq t \leq T$ and $i \in [m]$.

**Proof.** The proof using convexity assumption as well as Lemmas 2, 3, and 4 is provided in Section 4.6.4. \qed

**Remark 4.** The convergence inequality for convex objectives in (4.36) has two terms. The first term is the standard convex optimization term, and the second term comes from the error in finding the solution of the inner level at each iteration of the outer level.

To achieve an $\epsilon$-accurate Pareto efficient solution for the outer level, we need to take $T = O\left(\frac{1}{\epsilon^2}\right)$ steps of gradient descent. In this setting the error of approximation for the inner level can be intensified by increasing the number of fairness objectives $m$. This is due to $A_\Psi = O\left(\sqrt{m}\right)$.

Whenever the objective functions of the problem are not convex, following the standard non-convex optimization, we have:
**Theorem 4** (Non-convex). Let \( h(w) = [h_1(w), \ldots, h_m(w)] \) be the objective vector that follows the properties in Assumptions 1.a, 1.b, and 1.c. Then the function \( \Psi(h(w), \alpha) \) is a smooth function with Lipschitz constant of \( L_\Psi \) from Lemma 4. Also, the inner function of optimization (4.5) has the properties in Assumptions 2.a, 2.b, and 2.c. Then, for the average squared norm of the gradient of the outer function in the sequence of solutions \( w^{(0)}, \ldots, w^{(T)} \), generated by Algorithm 2, by setting \( \eta \leq 1/L_\Psi \), we have

\[
\frac{1}{T} \sum_{t=0}^{T-1} \left\| \nabla \Psi(h(w^{(t)}), \alpha^*(w^{(t)})) \right\|^2 \\
\leq \frac{2}{\eta T} \left( \Psi(h(w^{(0)}), \alpha^*(w^{(0)})) - \Psi(h(w^*), \alpha^*(w^*)) \right) \\
+ \frac{L_\Psi \eta A^2_\Psi}{T} \exp \left( -\frac{K}{\kappa} \right) \sum_{t=0}^{T-1} \left\| \alpha^{(0)}(w^{(t)}) - \alpha^*(w^{(t)}) \right\|^2 (4.37)
\]

**Proof.** The proof using smoothness assumption of the outer level in Lemma 4, and also Lemmas 2, and 3 is deferred to the Section 4.6.5. \( \square \)

**Remark 5.** Using the convergence analysis of non-convex outer-level objective function in (4.37), we can bound the minimum squared norm of the its gradient to be less than \( \epsilon \), with taking \( O\left(\frac{1}{\epsilon^2}\right) \) gradient steps. Note that, when this gradient is zero, we are in a Pareto stationary point of the problem.

### 4.5 Final Remarks

In this chapter, we propose two Pareto descent algorithms that are generic and can be applied to any multiobjective optimization problem. Both of these algorithms are gradient-based, and hence, can potentially can be applied to any arbitrary objective value. The theoretical results guarantee that the solution of these algorithms belong to the Pareto stationary set of the problem. Using the procedure described for PB-PDO not only we can converge to a solution with a desired level of compromises between objectives, but also we can extract points from the Pareto frontier of the optimization problem, which can shed light on the trade-offs between those objectives.

In addition, an interesting future direction worthy of exploration. The proposed algorithm is based on gradient descent method. While the stochastic version is evident, thorough convergence analysis for such method is required.
4.6 Missing Proofs

In this section, the missing proof from the main body of this chapter is presented.

4.6.1 Proof of Theorem 1

Before formally proving the existence of Pareto efficient solution as stated in Theorem 1, we would like to further elaborate on its implications. We remark that the condition described in Theorem 1 is about whether a Pareto efficient solution exists for a problem. Hence, there might be some extreme cases, where this solution does not exist or is not good enough. For instance, when objectives are completely contradicting each others but they are bounded below, a Pareto efficient solution exists but can be far away from the optimal point of each objective. Noting the inherent conflict between different objectives, our ultimate goal is to find the optimal trade-off between them. Thus, in most cases, with bounded objectives, well-defined Pareto efficient solutions exist and can be extracted. The following example will brighten the condition in Theorem 1 with two extreme cases where a Pareto efficient solution does not exist, and where it exists, but not well-defined.

![Figure 4.7: Value of $h_1(w)$ versus $h_2(w)$ for different $w$s for arbitrary parameters in Example 2.](image)

**Example 2.** (a) Consider that we have two objectives to satisfy and the parameter space is a single dimension space $w \in \mathcal{W} \subseteq \mathbb{R}^1$. Then, consider that the objectives are linear functions in the form of $h_1(w) = aw + b$, and $h_2(w) =cw + d$, where $a, b, c,$ and $d$ are
constants with $a,b > 0$. Next, the relationship between two objectives are:

$$h_2(w) = \frac{c}{a}h_1(w) + \frac{ad - bc}{a}.$$ 

Since, both the objectives are not bounded, when we want to minimize both together, we are not able to find a point that cannot be dominated by other points. This situation is depicted in Figure 4.7(a), for $a = 2$, $b = 3$, $c = 4$, and $d = -2$. As it can be inferred, for every chosen point like $w_1$ there exists a $\hat{w}$ that dominates it. Meaning, for objective vector of $h(w) = [h_1(w), h_2(w)]^\top$, we have $h(\hat{w}) \prec_m \ldots \prec_m h(w_3) \prec_m h(w_2) \prec_m h(w_1)$; hence, the set of Pareto efficient solutions would be empty.

(b) Now, consider that we have two objectives in a single dimension parameter space with the form of $h_1(w) = w^2$ and $h_2(w) = 1/w^2$. This means that both objectives are bounded below, but completely contradicting each other. In terms of the objectives, they have the relationship of:

$$h_2(w) = \frac{1}{h_1(w)}, \quad h_1(w) > 0 \quad \forall w \in \mathcal{W}.$$ 

In this extreme case, all the points belongs to the Pareto frontier because they cannot dominate each other as depicted in Figure 4.7(b). Thus, the Pareto efficient solution set is not well-defined.

4.6.1.1 Proof of Theorem 1

Proof. To prove the existence of a Pareto efficient solution, we define an auxiliary function and establish the conditions that guarantee the existence of the optimal solution. To this end, consider the following function defined at any point $\bar{w} \in \mathcal{W}$:

$$\Omega (w; \bar{w}) \triangleq \sum_{i=1}^{m} h_i (w)$$

s.t. $h(w) \prec_m h(\bar{w})$. \hfill (4.38)

Obviously, any feasible solution to $\arg \min_w \Omega (w; \bar{w})$ yields a solution that is as good as $\bar{w}$. Therefore, solving the above optimization results in solutions (if any exists) that are better than $\bar{w}$.

Now, we turn to characterizing the existence of a Pareto efficient solution as optimal solutions of the auxiliary problem defined in (4.38). First, it is easy to see by a simple
contradictory argument that any *efficient* solution $w^*$ should be an optimal solution to 
$\arg \min_w \Omega (w; w^*)$, i.e., $w^* \in \arg \min_w \Omega (w; w^*)$, with function value of $\sum_{i=1}^m h_i (w^*)$.

Of course, this can be utilized as an efficiency check (i.e., any efficient solution $w^*$ should belong to solution set of auxiliary function defined at that point $\arg \min_w \Omega (w; w^*)$). We also note that, if $w^*$ is an optimal solution to 
$\Omega (w; \bar{w})$, for a given point $\bar{w}$, then $w^*$ is efficient and we have:

$$ h_i (w^*) \leq h_i (\bar{w}), \quad i = 1, \ldots, m. \tag{4.39} $$

Therefore, if $\bar{w}$ is not in the optimal solution set of $\Omega (w; \bar{w})$, then, any optimal solution to $\Omega (w; \bar{w})$ would be efficient and better than $\bar{w}$, i.e., $\sum_{i=1}^m h_i (w^*) \leq \sum_{i=1}^m h_i (\bar{w})$. This can be easily shown by contradiction.

Equipped with the above auxiliary function and understanding the basic facts about its solution set, we only left with finding suitable conditions that guarantees the existence of optimal solutions of $\Omega (w; \bar{w})$. Indeed, for any $\bar{w} \in \mathcal{W}$, the minimization of the function in (4.38) has an optimal solution, i.e. $\bar{w}$, if the individual functions $h_i(\cdot), \ i = 1, \ldots, m$ are lower-semicontinuous over $\mathcal{W}$, and if the set $\{ w \in \mathcal{W}; h_i (w) \leq h_i (\bar{w}) , \forall 1 \leq i \leq m \}$ is compact. This holds when the constraint set is compact and the functions are continuous or bounded below as stated in the theorem.\footnote{We note that, an arbitrary function $h : \mathbb{R}^d \to [-\infty, +\infty]$ is lower-semicontinuous throughout $\mathbb{R}^d$ if the set $\{x|h(x) \leq \epsilon\}$ is closed for every $\epsilon \in \mathbb{R}$ or the epigraph of $h$ is closed set in $\mathbb{R}^{d+1}$. Obviously this condition holds for any convex function.}

We note that, when $\Omega (w; \bar{w})$ is unbounded below (i.e., achieving value $-\infty$), or it is finite but not attainable by any feasible solution, then no optimal solution exists. To see the former case, we consider the case that $\Omega (w; \bar{w}) = -\infty$ for some $w, \bar{w} \in \mathcal{W}$. Now, using the optimization problem of the function in (4.38), we write its dual problem as:

$$ d(\bar{w}) = \sup_{\lambda \in \mathbb{R}^m} \inf_{w \in \mathcal{W}} \sum_{i=1}^m h_i (w) + \lambda^\top (h(w) - h(\bar{w})) \tag{4.40} $$

where $\lambda$ is the dual parameter. Based on the weak duality theorem we know that we will have $d(\bar{w}) \leq h_i (\bar{w})$ for every $\bar{w}$. Then, if we have the objectives are not bounded below, we can have $h_i (\bar{w}) = -\infty$ for some $\bar{w} \in \mathcal{W}$, which implies that the dual problem also achieves a $-\infty$ and becomes infeasible. However, as long as the primal problem is feasible, a duality gap exists which in turn implies the efficiency of primal solution– meaning that the primal solution is Pareto efficient by the definition of auxiliary optimization problem.

Putting all together, we conclude that when the individual functions are bounded below and the constraint set is compact, then the solution set of auxiliary function is
non-empty which immediately implies the existence of an efficient solution. We note that these conditions can be relaxed to include other family of vector-valued optimization problems, but here we only focused on the characteristics that govern the optimization problem for finding a Pareto efficient solution.

4.6.2 Proof of Lemma 3

Proof. We start by expanding the error term for every \( \mathbf{w} \) and its corresponding \( \hat{\alpha} = \hat{\alpha}(\mathbf{w}) \) and \( \alpha^* = \alpha^*(\mathbf{w}) \) as follows. For brevity we replace \( \Psi(h(\mathbf{w}), \alpha(\mathbf{w})) \) and \( \Phi(h(\mathbf{w}), \alpha(\mathbf{w})) \) with \( \Psi(\mathbf{w}, \alpha) \) and \( \Phi(\mathbf{w}, \alpha) \), respectively, and their gradients accordingly.

\[
\| \nabla \Psi(\mathbf{w}, \hat{\alpha}) - \nabla \Psi(\mathbf{w}, \alpha^*) \| = \| \nabla_\mathbf{w} \Psi(\mathbf{w}, \hat{\alpha}) + \nabla_{\hat{\alpha}} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_\mathbf{w} \Psi(\mathbf{w}, \alpha^*) - \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*) \|
\leq \| \nabla_\mathbf{w} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_\mathbf{w} \Psi(\mathbf{w}, \alpha^*) \|
+ \| \nabla_{\hat{\alpha}} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*) \|
+ \| (\nabla_{\hat{\alpha}} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*)) \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*) \|.
\]

Then, using the linear relationship of the outer level function \( \Psi(\mathbf{w}, \alpha) = \alpha^\top \mathbf{h} \) and bounded gradient in Assumption 1.b, we have:

\[
\| \nabla_\mathbf{w} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_\mathbf{w} \Psi(\mathbf{w}, \alpha^*) \| = \| \mathbf{G} \hat{\alpha} - \mathbf{G} \alpha^* \|
\leq \| \hat{\alpha} - \alpha^* \| \| \mathbf{G} \|_F
\leq B_h \sqrt{m} \| \hat{\alpha} - \alpha^* \|.
\]  

(4.41)

Replacing (4.41) in (4.41), we have:

\[
\| \nabla \Psi(\mathbf{w}, \hat{\alpha}) - \nabla \Psi(\mathbf{w}, \alpha^*) \| \leq B_h \sqrt{m} \| \hat{\alpha} - \alpha^* \|
+ \| \nabla_{\hat{\alpha}} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*) \|
+ \| (\nabla_{\hat{\alpha}} \Psi(\mathbf{w}, \hat{\alpha}) - \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*)) \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*) \|.
\]  

(4.42)

Again we use the linear relationship of the outer level, and hence, \( \nabla_{\alpha} \Psi(\mathbf{w}, \hat{\alpha}) = \nabla_{\alpha} \Psi(\mathbf{w}, \alpha^*) = \mathbf{h} \), which makes the second term in (4.42) zero and for the third term we use the Assumption 1.a on the function values and have \( \| \mathbf{h} \| \leq D_h \). Also replace \( \nabla_{\hat{\alpha}}(\mathbf{w}) \)
and $\nabla \alpha^*(w)$ with approximation in (4.33), we will have:

$$
\| \nabla \Psi(w, \hat{\alpha}) - \nabla \Psi(w, \alpha^*) \|
\leq B_h \sqrt{m} \| \hat{\alpha} - \alpha^* \|
+ D_h \sqrt{m} \left\| \nabla^2_{w\alpha} \Phi (w, \hat{\alpha}) \left( \nabla^2_{\alpha\alpha} \Phi (w, \hat{\alpha}) \right)^{-1} - \nabla^2_{w\alpha} \Phi (w, \alpha^*) \left( \nabla^2_{\alpha\alpha} \Phi (w, \alpha^*) \right)^{-1} \right\|_F
\leq B_h \sqrt{m} \| \hat{\alpha} - \alpha^* \|
+ D_h \sqrt{m} \left\| \nabla^2_{w\alpha} \Phi (w, \hat{\alpha}) \left( \left[ \nabla^2_{\alpha\alpha} \Phi (w, \hat{\alpha}) \right]^{-1} - \left[ \nabla^2_{\alpha\alpha} \Phi (w, \alpha^*) \right]^{-1} \right) \right\|_F. \quad (4.43)
$$

Next, we use the fact that $\nabla^2_{\alpha\alpha} \Phi (w, \alpha^*) = \nabla^2_{\alpha\alpha} \Phi (w, \hat{\alpha}) = G^T G$, which makes the last term zero. Also, using Assumption 2.a, we have the strong convexity property of the inner problem $\|G^T G\| \geq \mu_\Phi$; and the Lipschitz continuity of $\nabla^2_{w\alpha} \Phi (w, \alpha)$ from Assumption 2.c:

$$
\| \nabla \Psi(w, \hat{\alpha}) - \nabla \Psi(w, \alpha^*) \|
\leq B_h \sqrt{m} \| \hat{\alpha} - \alpha^* \|
+ D_h \sqrt{m} \left\| \nabla^2_{w\alpha} \Phi (w, \hat{\alpha}) - \nabla^2_{w\alpha} \Phi (w, \alpha^*) \right\|_F / \left\| \nabla^2_{\alpha\alpha} \Phi (w, \alpha^*) \right\|_F
\leq \left( B_h \sqrt{m} + \frac{D_h \sqrt{m}}{\mu_\Phi} L_{w\alpha} \right) \| \hat{\alpha} - \alpha^* \|. \quad (4.44)
$$

\[\Box\]

4.6.3 Proof of Lemma 4

Proof. It is straightforward from the definition of $\Psi(w, \alpha)$ in (4.5). We start by expanding the term using the definition of the $\Psi$ function. For brevity we replace $\Psi(h(w), \alpha(w))$ and $\Phi(h(w), \alpha(w))$ with $\Psi(w, \alpha)$ and $\Phi(w, \alpha)$, respectively, and their gradients accordingly:

$$
\| \nabla \Psi(w_1, \alpha^*(w_1)) - \nabla \Psi(w_2, \alpha^*(w_2)) \|
= \| G \alpha^*(w_1) - G \alpha^*(w_2) \|
= \| (G(w_1) - G(w_2)) \alpha^*(w_1) + G(w_2) (\alpha^*(w_1) - \alpha^*(w_2)) \|
\leq \| (G(w_1) - G(w_2)) \alpha^*(w_1) \| + \| G(w_2) (\alpha^*(w_1) - \alpha^*(w_2)) \|
\leq \| \alpha^*_1 (w_1) (g_1(w_1) - g_1(w_2)) + \ldots + \alpha^*_m (w_1) (g_m(w_1) - g_m(w_2)) \|.
$$

63
\[ \|G(w_2)\|_F \|\alpha^*(w_1) - \alpha^*(w_2)\|. \quad (4.45) \]

Then, we use the triangle inequality for the first term. As for the second term, we use the definition in (4.33) to bound the gradient of the inner level variable as:

\[
\|\nabla \alpha^*(w)\| = \left\| \nabla^2_{\alpha \alpha} \Phi(w, \alpha^*(w)) \left[ \nabla^2_{\omega \alpha} \Phi(w, \alpha^*(w)) \right]^{-1} \right\|
\leq \left\| \nabla^2_{\omega \alpha} \Phi(w, \alpha^*(w)) \right\| / \left\| \nabla^2_{\alpha \alpha} \Phi(w, \alpha^*(w)) \right\|
\leq H_\Phi / \mu_\Phi, \quad (4.46)
\]

where we use the Assumptions 2.c and 2.a for bounding of \( \nabla^2_{\omega \alpha} \Phi(w, \alpha^*) \) and strong convexity parameter of \( \nabla^2_{\alpha \alpha} \Phi(w, \alpha^*) \). This implies that \( \alpha^*(w) \) is Lipschitz continuous with the constant of \( H_\Phi / \mu_\Phi \). Plugging back (4.46) into (4.45), we will have:

\[
\|\nabla \Psi(w_1, \alpha^*(w_1)) - \nabla \Psi(w_2, \alpha^*(w_2))\| \leq \alpha_1^*(w_1) \|g_1(w_1) - g_1(w_2)\| + \ldots
+ \alpha_m^*(w_1) \|g_m(w_1) - g_m(w_2)\|
+ B_h \sqrt{m} \cdot \frac{H_\Phi}{\mu_\Phi} \|w_1 - w_2\|. \quad (4.47)
\]

Now, we use the Assumption 1.c, for smoothness of each objective function with their respective parameter \( L_i, i \in [m] \):

\[
\|\nabla \Psi(w_1, \alpha^*(w_1)) - \nabla \Psi(w_2, \alpha^*(w_2))\| \leq \alpha_1^*(w_1) \|w_1 - w_2\| + \ldots
+ L_m \alpha_m^*(w_1) \|w_1 - w_2\|
+ \frac{H_\Phi B_h \sqrt{m}}{\mu_\Phi} \|w_1 - w_2\|. \quad (4.48)
\]

Finally, we replace each smoothness parameter with \( L_{\max} \triangleq \max \{L_1, \ldots, L_m\} \) and note that \( \alpha^*(w) \in \Delta_m \), we will have:

\[
\|\nabla \Psi(w_1, \alpha^*(w_1)) - \nabla \Psi(w_2, \alpha^*(w_2))\| \leq \left( L_{\max} + \frac{H_\Phi B_h \sqrt{m}}{\mu_\Phi} \right) \|w_1 - w_2\| \quad (4.49)
\]

\[\Box\]
4.6.4 Proof of Theorem 3

Equipped with Lemma 3, we can now turn to the proof of convergence for the convex outer objectives:

**Proof.** Following the convexity of each objective function of $f_i$, we can write the following inequality:

$$h_i(w(t)) - h_i(w^*) \leq \nabla h_i(w(t))^\top (w(t) - w^*)$$  \hspace{1cm} (4.50)

where $w^*$ is the solution to problem (4.5). Considering that the weights for each objectives $\alpha_i$ is always non-negative values, their weighted sum is also convex and we have:

$$\Psi(w(t), \hat{\alpha}(w(t))) - \Psi(w^*, \alpha^*(w^*)) \leq \nabla_w \Psi(w(t), \hat{\alpha}(w(t)))^\top (w(t) - w^*)$$

where, $\Psi$ comes from the update rule of the outer level. Now, from smoothness of the outer function from Lemma 4, we have:

$$\Psi(w(t), \alpha^*(w(t))) \leq \Psi(w(t), \alpha^*(w(t))) + \nabla \Psi(w(t), \alpha^*(w(t)))^\top (w(t+1) - w(t))$$

$$+ L_{w} \frac{1}{2} \|w(t+1) - w(t)\|^2.$$  \hspace{1cm} (4.52)

Then, by plugging the inequality of (4.51) in (4.52) and the fact that $\Psi(w(t), \alpha^*(w(t))) \leq \Psi(w(t), \hat{\alpha}(w(t)))$, we will have:

$$\Psi(w(t+1), \alpha^*(w(t+1))) \leq \Psi(w^*, \alpha^*(w^*)) + \nabla \Psi(w(t), \alpha^*(w(t)))^\top (w(t+1) - w(t))$$

$$+ L_{w} \frac{1}{2} \|w(t+1) - w(t)\|^2 + \frac{1}{2\eta} \left(\|w(t) - w^*\|^2 - \|w(t+1) - w^*\|^2\right)$$

$$+ \left(\frac{1}{2\eta} + \frac{L_{w}}{2}\right) \|w(t+1) - w(t)\|^2.$$  \hspace{1cm} (4.53)

Then, we are adding and subtracting the inexact gradient term of $\nabla \Psi(w(t), \hat{\alpha}(w(t)))$. Also, we use the update rule of the outer level, and denote the gradient error by...
\[ \Lambda(t) \triangleq \nabla \Psi \left( w(t), \hat{\alpha} \left( w(t) \right) \right) - \nabla \Psi \left( w(t), \alpha^* \left( w(t) \right) \right) \] for every iteration of the outer level, hence, we will have:

\[
\Psi \left( w^{(t+1)}, \alpha^* \left( w^{(t+1)} \right) \right) \\
\leq \Psi \left( w^*, \alpha^* \left( w^* \right) \right) \\
+ \langle \nabla \Psi \left( w(t), \hat{\alpha} \left( w(t) \right) \right) - \nabla \Psi \left( w(t), \hat{\alpha} \left( w(t) \right) \right), w^{(t+1)} - w(t) \rangle \\
+ \langle \nabla \Psi \left( w(t), \hat{\alpha} \left( w(t) \right) \right)^T \left( w^{(t+1)} - w(t) \right) \\
+ \frac{1}{2\eta} \left( \| w(t) - w^* \|^2 - \| w(t+1) - w^* \|^2 \right) \\
+ \left( \frac{1}{2\eta} + \frac{L_\Psi}{2} \right) \| w^{(t+1)} - w(t) \|^2 \\
= \Psi \left( w^*, \alpha^* \left( w^* \right) \right) + \langle \Lambda(t), w(t) - w^{(t+1)} \rangle \\
+ \frac{1}{2\eta} \left( \| w(t) - w^* \|^2 - \| w(t+1) - w^* \|^2 \right) \\
- \eta \| \nabla \Psi \left( w(t), \hat{\alpha} \left( w(t) \right) \right) \|^2 \\
+ \left( \frac{1}{2\eta} + \frac{L_\Psi}{2} \right) \eta^2 \| \nabla \Psi \left( w(t), \hat{\alpha} \left( w(t) \right) \right) \|^2. \tag{4.54}
\]

Next, we set the outer level learning rate to be \( \eta \leq 1/L_\Psi \) and use the fact that \( \langle a, b \rangle \leq \| a \| \| b \| \) and \( R = \max_w \| w_1 - w_2 \| \) for any \( w_1, w_2 \in \Omega \):

\[
\Psi \left( w^{(t+1)}, \alpha^* \left( w^{(t+1)} \right) \right) \leq \Psi \left( w^*, \alpha^* \left( w^* \right) \right) \\
+ \frac{1}{2\eta} \left( \| w(t) - w^* \|^2 - \| w(t+1) - w^* \|^2 \right) + R \| \Lambda(t) \|. \tag{4.55}
\]

Finally, we are use Lemma 2 and 3 to bound the gradient error:

\[
\Psi \left( w^{(t+1)}, \alpha^* \left( w^{(t+1)} \right) \right) \leq \Psi \left( w^*, \alpha^* \left( w^* \right) \right) + \frac{1}{2\eta} \left( \| w(t) - w^* \|^2 - \| w(t+1) - w^* \|^2 \right) \\
+ RA_\Psi \exp \left( -\frac{K}{2\eta} \right) \| \alpha^{(0)} \left( w(t) \right) - \alpha^* \left( w(t) \right) \|. \tag{4.56}
\]

Then, by summing up both sides over \( t \) from 0 to \( T - 1 \) and dividing by \( T \), and by denoting \( \bar{w} = \sum_{t=1}^{T} w(t)/T \) and \( w^{T+1} = w^* \), we have:

\[
\Psi \left( w, \bar{\alpha} \right) - \Psi \left( w^*, \alpha^* \left( w^* \right) \right) \leq \frac{1}{2\eta T} \| w^{(0)} - w^* \|^2
\]
Then, using the definition of $\bar{\alpha} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_m]$, and $\tilde{\alpha}_i = \min_t \alpha^*_i (w(t))$ for $1 \leq t \leq T$ and $i \in [m]$.

4.6.5 Proof of Theorem 4

Proof. When the main function is generally non-convex, we can follow the convergence analysis with a modification for the residual error because of the inner level approximation. Note that, the inner level function is strongly convex, and hence, the results from Lemma 3 are valid. Now, by considering the smoothness of the outer level from Lemma 4 and the update rule of the outer level, we can write:

$$
\Psi \left( w^{(t+1)}, \alpha^* \left( w^{(t+1)} \right) \right) \leq \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) + \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right)^\top \left( w^{(t+1)} - w^{(t)} \right)
+ \frac{L \eta}{2} \left\| w^{(t+1)} - w^{(t)} \right\|^2
= \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right)
- \eta \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right)^\top \nabla \Psi \left( w^{(t)}, \alpha \left( w^{(t)} \right) \right)
+ \frac{L \eta \eta}{2} \left\| \nabla \Psi \left( w^{(t)}, \alpha \left( w^{(t)} \right) \right) \right\|^2.
$$

(4.58)

Then, using the definition of $\Lambda^{(t)}$ from Appendix 4.6.4, we can write:

$$
\Psi \left( w^{(t+1)}, \alpha^* \left( w^{(t+1)} \right) \right) \leq \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) - \eta \left\| \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) \right\|^2
- \eta \left\langle \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right), \Lambda^{(t)} \right\rangle
+ \frac{L \eta \eta}{2} \left\| \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) + \Lambda^{(t)} \right\|^2
= \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) + \left( \frac{L \eta \eta}{2} - \eta \right) \left\| \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) \right\|^2
+ \frac{L \eta \eta}{2} \left\| \Lambda^{(t)} \right\|^2
- \left( \eta - L \eta \eta \right) \left\langle \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right), \Lambda^{(t)} \right\rangle.
$$

(4.59)

Next, by setting the learning rate $\eta \leq 1/L \eta$, we have:

$$
\Psi \left( w^{(t+1)}, \alpha^* \left( w^{(t+1)} \right) \right) \leq \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) + \left( \frac{L \eta \eta}{2} - \eta \right) \left\| \nabla \Psi \left( w^{(t)}, \alpha^* \left( w^{(t)} \right) \right) \right\|^2
$$

67
\[ + \frac{L\psi\eta^2}{2} \| \Lambda(t) \|^2 \] (4.60)

Finally, we use Lemmas 2 and 3 to bound the gradient error of the outer level:

\[
\Psi \left( w^{(t+1)}, \alpha^*(w^{(t+1)}) \right) \leq \Psi \left( w^{(t)}, \alpha^*(w^{(t)}) \right) + \left( \frac{L\psi\eta^2}{2} - \eta \right) \| \nabla \Psi \left( w^{(t)}, \alpha^*(w^{(t)}) \right) \|^2 \\
+ \frac{L\psi\eta^2}{2} \cdot A_{\psi}^2 \exp \left( -\frac{K}{\kappa} \right) \| \alpha^{(0)}(w^{(t)}) - \alpha^*(w^{(t)}) \|^2.
\] (4.61)

Now, by rearranging the terms and summing both sides over \( t \) we will have:

\[
\eta \left( 1 - \frac{L\psi\eta}{2} \right) \sum_{t=0}^{T-1} \| \nabla \Psi \left( w^{(t)}, \alpha^*(w^{(t)}) \right) \|^2 \\
\leq \Psi \left( w^{(0)}, \alpha^*(w^{(0)}) \right) - \Psi \left( w^*, \alpha^*(w^*) \right) \\
+ \frac{L\psi\eta^2 A_{\psi}^2}{2} \exp \left( -\frac{K}{\kappa} \right) \sum_{t=0}^{T-1} \| \alpha^{(0)}(w^{(t)}) - \alpha^*(w^{(t)}) \|^2
\] (4.62)

Then, considering the choice for \( \eta \), which makes \( \eta \left( 1 - \frac{L\psi\eta}{2} \right) \geq \frac{\eta}{2} \), and divide the both sides of (4.62) by \( T \), we have:

\[
\frac{1}{T} \sum_{t=0}^{T-1} \| \nabla \Psi \left( w^{(t)}, \alpha^*(w^{(t)}) \right) \|^2 \leq \frac{2}{\eta T} \left( \Psi \left( w^{(0)}, \alpha^*(w^{(0)}) \right) - \Psi \left( w^*, \alpha^*(w^*) \right) \right) \\
+ \frac{L\psi\eta^2 A_{\psi}^2}{T} \exp \left( -\frac{K}{\kappa} \right) \sum_{t=0}^{T-1} \| \alpha^{(0)}(w^{(t)}) - \alpha^*(w^{(t)}) \|^2
\] (4.63)

\[ \Box \]
As algorithmic decision-making systems are becoming more pervasive, it is crucial to ensure such systems do not become mechanisms of unfair discrimination on the basis of gender, race, ethnicity, religion, etc. Moreover, due to inherent trade-off between fairness measures and accuracy, it is desirable to learn fairness-enhanced models without significantly compromising the accuracy. In this chapter, we propose Pareto efficient Fairness (PEF) as a suitable fairness notion for supervised learning, that can ensure the optimal trade-off between overall loss and other fairness criteria. The proposed PEF notion is definition-agnostic, meaning that any well-defined notion of fairness can be reduced to the PEF notion. To efficiently find a PEF classifier, we use the proposed Pareto descent optimization algorithm in Chapter 4 that can guarantee the solution belongs to the Pareto frontier with provable guarantees for convex and non-convex objectives. We also apply the proposed algorithm for extracting the Pareto frontier of multiobjective optimization to find the accuracy-fairness Pareto frontier. We empirically demonstrate the effectiveness of PEF solution and the extracted Pareto frontier on real-world datasets compared to state-of-the-art methods.

5.1 Introduction

Attracting momentous attention during the past few years, machine learning models are significantly impacting nearly every aspect of the modern society we live in. Hence, it is of paramount importance to study how these models are affecting our lives, and to what
extent they are upholding to the moral standards of our society. The degree of fairness in algorithmic decision-making systems has been the center of a heated debate in numerous fields, such as criminal justice, advertising, admissions, and hiring [5]. Recently, the concerns about algorithmic fairness have resulted in a resurgence of interest to develop fairness-aware predictive models to ensure such models do not become a source of unfair discrimination on the basis of gender, race, ethnicity, religion, etc. Devising a fairness-enhanced mechanism necessitates i) a precise fairness measure to quantify fairness by taking into account the proper legal, ethical, and social context, and ii) developing an algorithmic solution to learn models that guarantee such metrics during test time.

Inspired by legal notions of fairness, a great deal of efforts has been invested in defining the right notion of algorithmic fairness such as demographic (or statistical) parity, equalized odds and equality of opportunity [64], individual fairness [41], representational fairness [156], and fairness under composition [40]. Yet, there is a lack of general consensus on which definition can best satisfy fairness requirements as different measures exhibit different advantages and disadvantages. Interestingly, several studies suggest that some of these definitions are incompatible with each other [126].

From an algorithmic standpoint, we seek to design machine learning algorithms that yield predictive models, which are demonstrably fair and bolster to alleviate the impact of unfair decisions [64, 153, 155, 38, 126]. This can be accomplished by changing the learning procedure, whether in pre-processing, in-process training, or post-processing stages [12]. Most of existing algorithmic or in-process approaches mainly aim at solving a constrained optimization problem by imposing a constraint on the level of fairness while optimizing the main learning objective, e.g., accuracy [38, 149, 155]. Furthermore, to manage non-convex constrains resulted from fairness constrains, these methods mostly employ some relaxations, such as linear relaxation such as FERM [38], to make the optimization problem more tractable. However, due to inherent trade-off between accuracy and fairness which is demonstrated in recent studies both empirically and theoretically [88, 101, 155], simply imposing fairness constraint to the main learning objective may significantly compromise accuracy. That is because there is no such thing as a free lunch, meaning, imposing fairness constraints to the main learning task, introduces a trade-off between these objectives. These trade-offs between the fairness constraints and accuracy has been asserted in several studies [88, 101, 155, 81, 147].

In light of incompatibility of fairness measures, and inherent trade-off between accuracy and fairness— as perusing higher degree of fairness will compromise accuracy, a fundamental research question is: for a given fairness measure, can we learn a model
that allows for higher fairness without significantly compromising accuracy? And a more important question is: Can we find a set of solutions with different levels of trade-off between accuracy and fairness, so a decision-maker can choose from depending on the preference of one objective over the other in an application?

Answers to these question need to be compatible with the fairness measure while minimizing the price of fairness, i.e., the relative reduction in the accuracy under the fair solution compared to the best accuracy without fairness consideration. Dealing with multiple and possibly conflicting objectives, a conspicuous approach is to seek a Pareto optimal solution to guarantee optimal compromises between accuracy and fairness. A model is Pareto optimal if there is no alternative model to make one of the objectives (accuracy or fairness) better off without making the other worse.

The main practical difficulty of the aforementioned constrained-based minimization approaches such as FERM or reductions [2] is that in general the obtained solution might not be a Pareto optimal solution. As a result, the optimal trade-off between fairness measure and accuracy is not guaranteed. Moreover, most of state-of-the-art algorithms for fairness have been proposed for binary sensitive (i.e., protected) features in a binary classification problems; however, the generalization of these methods to multiple group sensitive features, or multi-label classification tasks are limited or computationally inefficient [2, 38, 153, 134]. For example, Figure 5.1 shows that FERM algorithm leads to unnecessary sacrifices of overall accuracy to satisfy a fairness criteria (for more details see Section 5.4).

A remarkable attempt to achieve better accuracy-fairness trade-offs has been made in [2], where the fairness constrained optimization problem is cast as a saddle point problem, and an approximate equilibrium is sought by utilizing the exponentiated-gradient algorithm. However, the sequence of randomized classifiers returned by their algorithm requires solving a cost-sensitive classification at each step which is computationally burdensome. Also, the non-dominating solutions resulting from their algorithm are not necessarily from the Pareto frontier, and hence, they might not be Pareto efficient. Finally, the proposed grid-search reduction to learn a deterministic fair classifier from randomized classifiers is prohibitively costly for non-binary sensitive features.

To mitigate these issues, in this chapter, we introduce the notion of Pareto efficient fairness (PEF)\textsuperscript{1}, through which we indicate the optimal trade-off between fairness and accuracy. This notion is definition-agnostic, and encompasses many other previously

\footnotetext{\textsuperscript{1}Notion of PEF is also used in [10] for their specific fairness measure that is different from our definition of the PEF here. Our notion is more generic and is not bounded to any fairness measure. For more discussion refer to Section 5.2.}
Figure 5.1: Comparing the proposed Pareto efficient algorithm on COMPAS dataset [5] with normal linear SVM training and the proposed algorithm in [38], called fair empirical risk minimization (FERM). We consider “sex” as the sensitive feature. The results indicate the notion of Pareto efficient fairness can, indeed, achieve to the state-of-the-art results, and, at the same time, have the optimal compromises between total accuracy and equal true positive rates among groups. As it can be inferred, FERM sacrifices more accuracy (reflected in more false positive rate), in order to achieve higher and equal true positive rates. The details of experiments are discussed in Section 5.4.

studied definitions of fairness as special cases. Moreover, it is straightforward to adopt the proposed PEF notion to multiple group sensitive features or other machine learning problems beyond binary classification. To learn a PEF classifier, we use the bilevel Pareto descent optimization algorithm introduced in Chapter 4, which can be utilized by any classification model – which is trainable via gradient descent, to generate a fairness-enhanced model from Pareto frontier of the problem. Also using the novel first-order method in Algorithm 3, we are able to trace points from the Pareto frontier of accuracy-fairness trade-off. To the best of our knowledge, this is the first proposal that specifically designed to trace points on the Pareto frontier of the fairness-aware learning problem.

We rigorously analyze the convergence of the proposed algorithm to a Pareto optimal solution which guarantees the optimal accuracy-fairness trade-off.

Contributions. Below, we clarify the relationship and differences between our work and earlier research in fair classification. In particular, the main contributions of this research consists of proposing an algorithmic solution unifying and extending existing methods in several aspects:

- This proposal introduces a general framework for fairness-aware classification problem, that can be applied to different notions of fairness, with convergence guarantees to the Pareto stationary points of the optimization problem. Unlike prior methods for fairness-aware learning, the proposed framework does not employ
any relaxation assumptions for objectives in the optimization problem, hence it could achieve state-of-the-art results using a gradient descent method.

- Using a gradient descent approach makes the optimization process computationally efficient with respect to other state-of-the-art methods in this domain.
- Using the algorithmic solution we proposed we can trace points on the Pareto frontier of the fairness-accuracy trade-off with theoretical guarantees of Pareto stationary points.

### 5.2 Additional Related Work

Achieving a “fair” classifier is becoming the main propulsion of an expanding line of research, trying to define an all-inclusive definition for fairness and quantify it in the learning task. A notable attempt in this field by [64] introduced Equalized Odds and Equality of Opportunity in the context of supervised learning. However, there is not a consensus on the definition of fairness in algorithmic decision-making systems. In addition, there has been asserted that some of the current well-known definitions of fairness cannot be satisfied simultaneously [94]. Nevertheless, the efforts in this field, in general, can be categorized as either mitigating the effect of biases on protected groups or individuals in current decision-making systems, or modeling the long-term effect of the former methods [104, 85, 105]. In this research, we are dealing with the former category. Being rife with various notions and viewpoints of fairness, we summarize some key research in this field, related to our framework as follows.

**Individual Fairness**  One worldview of the fairness problem is that similar individuals in the input features’ space, should have similar outputs in the decision space [52]. Following this rule, [41] introduced the individual notion of fairness by using task-specific similarity metrics that can be enforced like a Lipschitz continuity constraint to the classifier. However, [52] introduce another worldview, called structural bias, in which it applies to many real-world societal applications. Most of the cases, the bias, culturally or historically, has affected a group of individuals, all having a similar set of characteristics. Hence, using a similarity metric to assure fairness in individual level would be a simplistic approach that cannot be effective when such a structural bias presents. That is why a great body of literature revolves around the problem of group fairness.
**Group Fairness**  The studies in the field of group fairness can be categorized in one of the three following strategies: The first approach is to pre-process data, either by modifying the sensitive features or map the data to a new space [38, 41, 48]. This methods are prone to fail, since they are not accuracy-driven and oblivious to the training procedure time. The second approach, tries to modify the existing classifiers, in order to impose the fairness constraints [126, 58, 23, 24]. The third approach, to which our proposed framework belongs, try to integrate the fairness constraints in the training [155, 149, 154, 111, 84, 2, 153]. Despite their success, they are mostly bound to binary classification or binary sensitive groups, which makes them impractical. Moreover, the efficiency of their solution is left as an open problem [155]. We propose a unified approach to address the efficiency of the solution, as well as, generalization to complex multiple group and multi-label classification tasks.

**Fairness-Accuracy Trade-offs**  The trade-offs between accuracy, as the main learning objective, and different fairness measures has been investigated widely in different studies. In [18], authors introduce two models for the bias source in data, namely, under-representation and labeling bias. They argue that the ERM with equality of opportunity constraint under these forms of biases can achieve the Bayes optimal classifier. Although [70] suggests that minimizing ERM constrained to fairness objectives might not satisfy the Pareto principle for group welfare, but the group welfare was not in the objective of the optimization problem in the first place. Hence, that is possible that a defined group welfare might not be satisfied with this optimization, unless its objective is added to the optimization. In [147], authors discuss that fairness objectives might be in accord with the main learning objective in a semi-supervised learning scenario, however, this is not generalizable to a supervised setting. Recently in [93], using confusion tensor of different notions of fairness in a binary classification, authors shows the trade-offs between predictive performance of a model and fairness objectives, as well as between fairness objectives themselves.

**Incompatibility of Notions**  Introducing numerous definitions and notions for fairness problem raise this critical question of which one is the most inclusive one. Despite the quest for finding the best notion for fairness, [94] and [26] discuss that some of the mainstream and widely used notions of fairness are incompatible; unless, there are some unrealistic assumptions about the classifier or data. Even if the notions are compatible, there might be trade-offs between different notions of fairness that make it impossible to
have them all satisfied at once, in addition to the trade-offs between these notions and accuracy.

**Pareto Fairness**  Dealing with the aforementioned trade-off between any fairness notion and the main learning objective, a conspicuous solution is to find the point that has the most desirable compromise between those objectives. Such a point is called Pareto efficient point, named after Italian economist Vilferdo Pareto. From the outset of this awareness on fairness measure in decision-making algorithms, finding a Pareto efficient point was an indisputable goal [155, 69, 10]. However, there has not been a unified framework to define the Pareto efficiency in the context of fairness problems, nor how to achieve it in general. Kearns & Roth in [88] in a chapter named "from Parity to Pareto" discuss why it is necessary to achieve a Pareto efficient point in this setting, rather than statistical parity. Perhaps the closest one to our proposal in finding the Pareto efficient solutions for fairness problems is [10]. However, their objectives are completely different from ours, where they consider two objectives and want to satisfy a balanced accuracy for each group with respect to their best accuracy they can achieve alone. Hence, they are finding a Pareto set, totally different from ours, by introducing a new notion for fairness. This measure, a.k.a. Chebychev measure, has been studied for unsupervised methods [81, 134], but it is not common in supervised ones, where the target values are available. On the other hand, our proposal is not introducing a new fairness notion, yet it can be applied to any existing fairness measure. Moreover, their proposed method is for an unbalanced skewed dataset with respect to different sensitive groups, which is only one form of biases in the fairness domain [18], while our approach is agnostic to the bias source. Finally, they do not provide any convergence analysis for their algorithm, which makes their approach more heuristic.

**Bilevel Optimization**  In this article, we lay the groundwork for Pareto efficiency in the fairness domain, and propose a proper multi-objective optimization. In order to solve it and guarantee the convergence to a Pareto efficient point, we will cast the problem as a bilevel programming. Bilevel programming is a well-known optimization framework that has two levels, inner and outer. In this setting, a solution of the inner problem is used to solve the outer problem. Bilevel optimization is based on a renowned two-player game, called Stackelberg game [13]. In this game, two players are called leader and follower, and they both want to minimize their specific objective functions. Recently, there has been a surge in the applications of this optimization problem, such as in hyperparameter
optimization [50, 120] or modeling different meta-learning approaches [80, 45, 129].

5.3 Pareto Efficient Fairness

In a typical fairness-aware supervised learning scenario, we have \( n \) i.i.d. training examples in the form of \( D = \{(x_1, a_1, y_1), \ldots, (x_n, a_n, y_n)\} \), where \( a_i \in A \), is a sensitive feature that represents group membership of each sample among \( c \) different groups, \( A = \{s_1, \ldots, s_c\} \). Our goal is to learn a function \( f : \mathcal{X} \mapsto \mathcal{Y} \) from input space \( \mathcal{X} \subseteq \mathbb{R}^d \) to output space \( \mathcal{Y} \) parametrized by a vector \( w \in \mathcal{W} \subseteq \mathbb{R}^d \). Note that, sensitive feature \( a \) might or might not be a part of the input feature \( x \). The performance of \( w \) is assessed using a loss function \( \ell : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}_+ \). For a normal training the empirical loss that is required to be minimized was defined in (4.1). Solely minimizing the empirical risk would result in an unfair solution with respect to different sensitive groups. To avoid that, researchers define constraints corresponding to each fairness notion they want to satisfy. The main idea to achieve a fair solution is to satisfy these fairness objectives in addition to the empirical risk of the problem.

In order to have a fair classifier, there are several different definitions that can be imposed to the learning process. These constraints try to reduce the effects of the sensitive feature \( a \) on the output of the classifier \( \hat{y} \). For instance, equality of opportunity introduced by [64] desires to ensure that the true positive rate of each sensitive group, \( TP_k = \mathbb{P} [\hat{y} = 1|a = s_k, y = 1] \), is the same for all \( k \in \{1, \ldots, c\} \). A stronger notion, equalized odds [64], requires that classifier’s output \( \hat{y} \) and sensitive feature \( a \) to be independent conditional on the true label \( y \). This reflects on not only having the same true positive rate among different groups, but also having an equal false positive rate for each group, \( FP_k = \mathbb{P} [\hat{y} = 1|a = s_k, y = -1] \) for all \( k \in \{1, \ldots, c\} \). Another notion, disparate mistreatment [154], calls for equal misclassification probability, \( F_k = \mathbb{P} [\hat{y} \neq y|a = s_k] \) for all \( k \in \{1, \ldots, c\} \).

Hence, as it was alluded to before, we ought to minimize several objectives simultaneously, where the objective vector in this case has the following form:

\[
\mathbf{w}^* = \arg \min_{w \in \mathcal{W}} \left\{ h(w) = [\mathcal{L}(w), h_2(w), \ldots, h_m(w)] \right\},
\]

(5.1)

where \( h_i(w), i = 2, \ldots, m \) are corresponding fairness violation objectives that ought to be minimized. As it was discussed in Chapter 4, we are seeking to find Pareto efficient points of this vector optimization problem. Hence, we call such points for this specific
vector optimization \textit{Paerto-Efficient Fair (PEF) solutions}.

### 5.3.1 Reduction of known notions of fairness to PEF

To further elaborate on PEF, we now show how different predefined notions of fairness such as equality of opportunity, equalized odds, and disparate mistreatment \cite{154} can be reduced to an instance of (5.1). Note that, we are not defining new notions of fairness, rather we are showing how we can reduce any notion of fairness to PEF.

#### 5.3.1.1 Pareto efficient equality of opportunity

Using Definition 2, we introduce a variant of equality of opportunity dubbed as \textit{Pareto efficient equality of opportunity}. For simplicity and adjusting to current state-of-the-art fairness definitions, we consider binary learning case, that is $Y = \{-1, +1\}$; however, as we denote later, the proposed algorithm can be simply generalized to multi-class learning. To satisfy equalized opportunity criteria, which is having the same true positive rate among different groups, we translate it as having equal loss on subset of samples of each group with positive labels, i.e. $S_k^+ = \{(x_i, a_i, y_i) \in D | a_i = s_k, y_i = 1, 1 \leq k \leq c\}$. This loss can be defined as:

$$L_k^+(w) = \frac{1}{|S_k^+|} \sum_{(x_i, y_i) \in S_k^+} \ell(w; (x_i, y_i)), \ k \in \{1, \ldots, c\}. \quad (5.2)$$

To achieve the equality of opportunity, we can use this empirical loss instead of the probability of positive outcome for each group using the classifier, as suggested by other studies \cite{38, 2}. In this scenario, to have equal probability, we try to have equal empirical loss for each group for positive outcome $L_k^+(w)$. Hence, the fairness objectives $H_{i,j}^+$ in this case reduces to the minimization of pairwise empirical losses between each pair of sensitive groups:

$$H_{i,j}^+(w) = \phi \left( L_i^+(w) - L_j^+(w) \right), \ 1 \leq i, j \leq c, i \neq j, \quad (5.3)$$

where $\phi(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^+$ is any penalization function, such as $\phi(z) = |z|$, $\phi(z) = \frac{1}{2}z^2$, or $\phi(z) = e^{-z}$, however, for convergence analysis we will stick to smooth ones, like squared or exponential penalization. Then, the objective vector for this fairness problem is $h_{EO}(w) = [L(w), H_{1,2}^+(w), \ldots, H_{c-1,c}^+(w)]^\top \in \mathbb{R}^m$, where $m = 1 + \binom{c}{2}$. A solution has the property of Pareto efficient equality of opportunity if it belongs to the PEF solution.
5.3.1.2 Pareto efficient equalized odds

As previously stated, satisfying equalized odds requires both true positive rates and false positive rates to be equal among all groups. We can use the same notion, to define its new variant, Pareto efficient equalized odds. For satisfying equal true positive rates, we use the same objectives as (5.3) for Pareto efficient equality of opportunity. To satisfy equal false positive rates among each group, we will, instead, focus on having equal true negative rate for each group, $\text{TN}_k = 1 - \text{FP}_k = \mathbb{P}[\hat{y} = -1|a = s_k, y = -1]$ for all $k \in \{1, \ldots, c\}$. This condition can be reflected using the corresponding loss, thus, we denote the set of samples for each group from negative class as $S_k^- = \{(x_i, a_i, y_i) \in \mathcal{D} | a_i = s_k, y_i = -1, 1 \leq k \leq c\}$. Then, the loss is:

$$L_k^-(w) = \frac{1}{|S_k^-|} \sum_{(x_i, y_i) \in S_k^-} \ell(w; (x_i, y_i)), \ k \in \{1, \ldots, c\}. \quad (5.4)$$

Using (5.4) and similarly to true positive rates, we can define pairwise objectives to be added as additional fairness objectives to the objective vector:

$$H_{i,j}^-(w) = \phi\left(L_i^-(w) - L_j^-(w)\right), \ 1 \leq i, j \leq c, i \neq j, \quad (5.5)$$

where $\phi(.)$, again, is the smooth penalization function, and the objective vector is $h_{\text{EOD}}(w) = [L(w), H_{1,2}^+(w), \ldots, H_{c-1,c}^+(w), H_{1,2}^-(w), \ldots, H_{c-1,c}^-(w)]^\top \in \mathbb{R}_+^m$, where number of objectives is $m = 1 + 2 \times \binom{c}{2}$. Similarly, a solution has the property of Pareto efficient equalized odds if it belongs to the PEF solution set of the constructed vector objective $h_{\text{EOD}}(w)$ in optimization of (5.1).

5.3.1.3 Pareto efficient disparate mistreatment

Analogous to two previously reduced notion of fairness to PEF, we can define Pareto efficient disparate mistreatment. In this notion, we need to have equal misclassification error rate among all groups in the sensitive feature. To that end, we denote the loss of each group as:

$$L_k(w) = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} \ell(w; (x_i, y_i)), \ k \in \{1, \ldots, c\}. \quad (5.6)$$
where $S_k = \{(x_i, a_i, y_i) \in D \mid a_i = s_k, 1 \leq k \leq c\}$ denotes each group’s set. Now, the fairness objectives for each pair of sensitive groups can be defined as:

$$H_{i,j}(w) = \phi (L_i(w) - L_j(w)), \ 1 \leq i, j \leq c, i \neq j,$$

(5.7)

to satisfy equal empirical loss on each group, in addition to the main loss. Thus, the objective vector is $h_{DM}(w) = [L(w), H_{1,2}(w), \ldots, H_{c-1,c}(w)]^\top \in \mathbb{R}_+^m$, where $m = 1 + \binom{c}{2}$. Following the previous definitions, a solution has the Pareto efficient disparate mistreatment effect if it belongs to the PEF solution set of the optimization in (5.1) using $h_{DM}(w)$.

### 5.4 Experimental Results

In this section, we empirically examine the efficacy of the proposed algorithms. The experiments are designed to answer the following questions:

1. How the proposed Pareto descent algorithm performs compared to a normal classifier in minimizing the empirical risk and fairness violation with a binary sensitive feature?

2. How the proposed Pareto descent algorithm performs compared to a normal classifier, when there is a multiple group sensitive feature?

3. How the proposed Pareto descent algorithm performs compared to state-of-the-art algorithms for fairness-aware learning, in particular the fairness constrained empirical risk minimization proposed in [38] and reduction to minimax optimization proposed in [2]?

4. How the proposed preference-based Pareto descent optimization can find solutions from different parts of the Pareto frontier? How it performs compared to state-of-the-art algorithm proposed in [2]?

In all the experiments, we conduct multiple runs, and report the average and the respective variance if applicable. To compare with both state-of-the-art algorithms (FERM [38] and reductions [2]) we run algorithms using linear SVM as the main objective function, since FERM is designed for this objective. In the third part, since we are finding the points from the Pareto frontier and only the reductions approach [2] claims to find such points, we only compare with their algorithm using a Logistic Regression as the
main objective function. Nonetheless, our proposed algorithms are not bounded to any objective functions and can be implemented in any setting.

**Datasets** We will use two real-world datasets: The Adult income dataset\(^2\) and the COMPAS dataset [5]. The meta-data for these datasets is provided in Appendix B. In the Adult dataset, the goal is to predict whether the income of each person is greater or less than $50K per year, based on census data. In the COMPAS dataset, the task is to predict whether the criminal defendant would recidivate within next two years or not based on historical data. In both datasets, the positive outcome (have income more than $50K per year in the Adult dataset or not recidivate within next two years in the COMPAS dataset) is considered beneficial. In Adult dataset, we have two sensitive features, **gender** and **race**, where gender is a binary feature, while race in this dataset is a multiple-group feature with 5 categories. In the COMPAS dataset, also, we have two sensitive features, **sex** and **race**, both of which are binary sensitive features.

1- **Binary sensitive feature: Pareto vs. Normal** In the first set of experiments, we examine the effectiveness of the proposed algorithm in satisfying the Pareto efficient equality of opportunity as defined in Section 5.3.1. Note that, under this condition the goal is to achieve equal true positive rates among sensitive groups. We apply Algorithm 2 to the Adult dataset with gender as the sensitive feature and also COMPAS dataset with race and sex as its sensitive features. The results are shown in Figure 5.2 for our algorithm compared to a normal training using a linear SVM, which indicates that the proposed algorithm can superbly satisfy the notion of equality of opportunity. As it can be inferred from the first column, in all three experiments the total (and also each sensitive group) accuracy for our proposed algorithm is almost the same as the normal training one. However, in the second column, we can see that our algorithm has equal true positive rates among sensitive groups, while the gap between different groups is huge for the normal training. This is achieved by the proposed algorithm, with almost the same total true positive rate as the normal training except for the first one (the Adult dataset). In this case, this degradation is compensated by smaller false positive rates for the proposed algorithm compared to the normal training. Also note that, the Pareto descent algorithm achieves equal false positive rates among sensitive groups, to some degree, which is not in the objectives of the equality of opportunity.

\(^2\)https://archive.ics.uci.edu/ml/datasets/Adult
Figure 5.2: The results of applying Algorithm 2 to satisfy the Pareto efficient equality of opportunity for (a) Adult dataset with gender as its sensitive feature, (b) COMPAS dataset with race as its sensitive feature (c) COMPAS dataset with sex as its sensitive feature. The drop in accuracy is small in all cases, while the goal of equal opportunity is satisfied perfectly. By looking at false positive rates, it seems that this notion is capable of satisfying equalized odds as well as equality of opportunity.

2- Multiple group sensitive feature: Pareto vs. Normal  In Figure 5.3, we show the results for applying the proposed algorithm to the Adult dataset with considering its multiple group sensitive feature, race. As it can be inferred, our algorithm can superbly satisfy the fairness constraints, while maintaining high accuracy close to the baseline. The figure in the middle shows that the Pareto descent algorithm achieves equal, and also higher, true positive rates among sensitive groups, however this comes at the price of slightly increasing false positive rates for them. Note that, for this case, we have 10 fairness objectives in addition to the main learning objective, which demonstrate the power of the proposed algorithm in finding an optimal point in terms of compromises between objectives. On the other hand, the applications of FERM [38] to multiple group sensitive features are not straightforward. Also, applying the reductions approach [2] to multiple group sensitive features is heavily computationally expensive.
Figure 5.3: Adult dataset with race as its sensitive feature, which has 5 groups. In this case, our proposed algorithm can superbly satisfy the property of Pareto efficient equality of opportunity, despite the fact that we have 11 objectives in the minimization. The groups are AIE: Amer-Indian-Eskimo, API: Asian-Pac-Islander, B: Black, O: Other, W: White, and T represents the total.

3- Pareto vs. state-of-the-arts From a more general perspective, the overarching goal of fairness-aware learning algorithms are to satisfy multiple objectives at the same time before, during, or after training a model. This view has been reflected as a constraint optimization problem in the algorithmic level in various approaches. In general, this optimization can be written as:

\[
\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}; \mathcal{D}) \quad \text{s.t.} \quad h_i(\mathbf{w}; \mathcal{D}) \leq \epsilon_i, \quad \forall i \in [m],
\]  

(5.8)

where \( h_i(\mathbf{w}; \mathcal{D}) \) is the \( i \)th fairness constraint we required to satisfy. In the quest for achieving fairness using constrained optimization in (5.8), two well-known studies by [38] and [2] introduced their frameworks using different methods. They attempted to solve this constrained optimization problem either by linear approximation of their constraints or forming the Lagrangian function to solve the saddle point minmax problem. In the latter, [2] used a grid search technique for binary sensitive feature problems in order to find the best weights for constraints in the Lagrangian objective. The greatest challenge in these approaches is to set the violation parameter \( \epsilon_i \) for each fairness constraint in the optimization. Also, finding the weights for each constraint in these approaches does not guarantee the Pareto efficiency of the solution. Not to mention the high computational demand of approaches such as grid search in [2].

Now, we can compare the results generated with Algorithm 2 with these two state-of-the-art algorithms for satisfying fairness measures, known as FERM [38] and reductions [2]. In order to compare with both, we run the experiments using linear SVM loss function since FERM is designed for this loss function. We also use the DEO to show fairness measures in addition to accuracy on test dataset. Table 5.1 summarizes the results we got from applying a normal linear SVM without fairness constraints, FERM linear SVM,
<table>
<thead>
<tr>
<th></th>
<th>Adult (gender)</th>
<th></th>
<th>COMPAS (sex)</th>
<th></th>
<th>COMPAS (race)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc</td>
<td>DEO</td>
<td>ACC</td>
<td>DEO</td>
<td>ACC</td>
<td>DEO</td>
</tr>
<tr>
<td>Normal</td>
<td>0.8123</td>
<td>0.2615</td>
<td>0.6037</td>
<td>±0.0058</td>
<td>0.2643</td>
<td>±0.0269</td>
</tr>
<tr>
<td>FERM [38]</td>
<td>0.7937</td>
<td>0.0173</td>
<td>0.5126</td>
<td>±0.0489</td>
<td>0.09337</td>
<td>±0.0724</td>
</tr>
<tr>
<td>Reductions [2]</td>
<td>0.7748</td>
<td>±0.0002</td>
<td>0.5805</td>
<td>±0.0047</td>
<td>0.0378</td>
<td>±0.0230</td>
</tr>
<tr>
<td>PDO (ours)</td>
<td>0.7969</td>
<td>±0.0003</td>
<td>0.5871</td>
<td>±0.0216</td>
<td>0.0354</td>
<td>±0.0293</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison of the proposed PDO algorithm with FERM [38] and reductions [2]. Experiments are repeated multiple times with reporting the average and variance. Blue colors show the best results for DEO measure, which in all cases belong to the proposed Pareto descent algorithm. Moreover, in terms of accuracy, the Pareto descent algorithm outperforms both state-of-the-art algorithms.

reductions on linear SVM, and our algorithm with gradient descent implementation of SVM. For the reductions algorithm [2], each grid search learns 100 different classifiers and the best non-dominated ones are selected. The numbers reported in Table 5.1 are the average of best non-dominate points for several grid search runs in terms of DEO. Overall, it shows that our algorithm can achieve a superb DEO compared to FERM and reductions, while having a better accuracy as well. The results show that in all three datasets our algorithm’s results dominate the solution of both state-of-the-art algorithms.

4- Pareto frontier: Pareto vs. Reductions [2] Using the proposed algorithm PB-PDO, we can extract the points from the Pareto frontier of the vector objective we are minimizing. In [2] also, authors claim that using their approach they can find some points from the Pareto frontier of the problem. However, in fact they are extracting some non-dominating points from multiple runs of their algorithm that are not necessarily on the Pareto frontier. In this part, we show how our algorithm performs on extracting the Pareto frontier and compare it to the points found by the reductions approach. We run the PB-PDO algorithm 10 times, for each dataset and each time with different preference vector, then we find the points that are not dominated in the trajectory. We set both \( \epsilon_1 \) and \( \epsilon_2 \) to \( 1e^{-2} \) for the PB-PDO in all algorithms. For the reductions, we run their algorithm to learn 200 classifiers and then find the non-dominating points for reporting.

We apply these algorithms on two loss functions: linear SVM and logistic regression. Figure 5.4 shows the results for applying both algorithms using linear SVM on the
Adults dataset with gender and COMPAS with sex and race as their sensitive features. Figure 5.5 shows the same results for both algorithms using logistic regression. In both figures, the first column is showing the trade-off between the main loss and the fairness loss (here equality of opportunity as defined in (5.3)). The proposed PB–PDO algorithm is training the Pareto frontier for this trade-off. Then, we map the points on this space (loss-loss) to the error versus DEO space. It is worth mentioning that this mapping is from a continuous space to a non-continuous one, and hence, we will loose some of the points as they map to the same point or getting dominated by other points in the error-deo space. This is exacerbated when the dataset is smaller like the COMPAS dataset. The second and the third columns are showing this mapping on train and test dataset, respectively. From the results in Figures 5.4 and 5.5 it is clear that almost all the points found by reductions approach as their non-dominated solutions, in fact, are dominated by the points found by our proposed Pareto descent optimization approach.

5.5 Final Remarks

In this chapter we advocate the notion of Pareto efficient fairness in dealing with fairness problems, and in order to achieve the optimal trade-offs between accuracy and other fairness criteria. By casting the fairness problem as a multi-objective optimization task and introducing Pareto descent optimizers as in Chapter 4, we can efficiently surpass existing methods for satisfying fairness criteria without significantly degrading the accuracy. We showed that other notions of fairness can be reduced to notion of Pareto fairness effortlessly, making the Pareto descent algorithm applicable to a range of fairness problems. Moreover, using our proposed framework finding PEF solutions from desired parts of the Pareto frontier of the problem is straightforward.

As future work to this chapter, it is interesting to understand the feasibility of different fairness criteria in the context of Pareto efficiency fairness, and how they are affecting the overall loss.
We map this Pareto frontier to the error vs. DEO trade-off for both training and test datasets on the second and the third columns, respectively. Our proposed algorithm, clearly, outperforms and dominates almost all the solutions found by the reductions approach.
Figure 5.5: Comparing the Pareto frontier extracted by our proposed algorithm PB-PDO and the reductions algorithm introduced in [2] using logistic regression as the loss function. We apply these algorithms on the Adult dataset with gender and the COMPAS dataset with sex and race as their sensitive features. The first column is the Pareto frontier on the trade-off between the main loss and the fairness loss (equality of opportunity as in (5.3)). We map this Pareto frontier to the error vs. DEO trade-off for both training and test datasets on the second and the third columns, respectively. Our proposed algorithm, clearly, outperforms and dominates almost all the solutions found by the reductions approach.
It has been shown that dimension reduction methods such as principle component analysis (PCA) may be inherently prone to unfairness and treat data from different sensitive groups such as race, color, sex, etc., unfairly. In pursuit of fairness-enhancing dimensionality reduction, using the notion of Pareto efficiency, we propose an adaptive first-order algorithm, similar to what we have in Chapter 4, to learn a subspace that preserves fairness, while slightly compromising the reconstruction loss. Theoretically, we provide sufficient conditions that the solution of the proposed algorithm belongs to the Pareto frontier for all sensitive groups; thereby, the optimal trade-off between overall reconstruction loss and fairness constraints is guaranteed. We also provide the convergence analysis of our algorithm and show its efficacy through empirical studies on different datasets, which demonstrates superior performance in comparison with state-of-the-art algorithms. The proposed fairness-aware PCA algorithm can be efficiently generalized to multiple group sensitive features and effectively reduce the unfairness decisions in downstream tasks such as classification.

6.1 Introduction

Recent advances in machine learning (ML) have vastly improved the capabilities of computational reasoning in complex domains. From tasks like image and video processing, game playing, text classification, to complex data analysis, machine learning is continually finding new applications and exceeding human-level performance in some
cases. Nevertheless, when machine learning models are trained on real data, the existing societal inequalities in data are manifested on the systems built upon them that could mislead models in ways that can have profound fairness implications such as being biased to sensitive features like race or gender. As more critical systems employ ML, such as financial systems, hiring and admissions, healthcare, and law, it is vitally important that we develop rigorous fair algorithms that are as accurate as possible.

Recently, the growing attention to the fairness problem in algorithmic decision-making systems has led to unprecedented attempts to revisit machine learning models for supervised and unsupervised tasks to satisfy fairness constraints [115]. An expanding line of works are dedicated to define different metrics for fairness problems and mechanisms to satisfy those measures in learning tasks such as [64, 155, 24, 23, 84, 2, 154, 153]. The work on this realm is focused on biased data or biased algorithms; however, using these biased algorithms in decision-making systems would lead to generating more biased data. This makes the causality of the fairness problem more complicated that exacerbates the problem even further [12, 57, 90].

Notwithstanding these efforts for fairness problem in supervised learning, fairness in unsupervised learning tasks has not been explored thoroughly. This is despite the fact that unsupervised learning tasks such as dimension reductions are mostly preceding those supervised ones, in the training procedures. Hence, having fair unsupervised learning models is as crucial as supervised ones. For instance, Principal Component Analysis (PCA) is widely used to reduce the dimension of the data before applying classification models. In addition to that, these unsupervised methods such as dimension reductions or clustering methods are commonly used for data visualizations, identifying common behaviors or trends, reducing the size of data, to name but a few. This ubiquitousness of unsupervised methods in machine learning models can affect decision-making systems if they unfairly treat different groups in data.

In this chapter, we aim at defining a fairness measure for dimension reduction algorithms like PCA and propose an algorithm to enforce these criteria in finding the subspace with minimum reconstruction loss. It is important to note that, despite supervised learning that fairness metrics are mostly focused on the beneficial outcome (usually the positive label), in an unsupervised task, there is no label to be used. Hence, we seek to find a subspace that is "good" enough for each protected group in the data. Indeed, when we apply PCA on a dataset, the resulting subspace found by a standard algorithm is different from what we achieve when only using the data of each group individually. This difference can be reflected as the difference between the reconstruction
Table 6.1: Comparison of time complexity of different fair PCA algorithms to achieve an \( \epsilon \)-fair subspace (please see Section 6.3 for definition). Here \( d \) denote the dimension of the original data, \( r \) is the target dimension, and \( k \) is the number of sensitive groups. Note that unlike previous studies that necessitates learning a subspace with larger dimension to guarantee fairness, our solution learns an exact \( r \) dimensional subspace by imposing additional constrains captured by a new notion of fairness proposed in this work to distinguish between local optimal fair subspaces (we used the following abbreviations above, SDP: Semi-Definite Programming, MW: Multiplicative Weight Algorithm, LP: Linear Programming, GD: Gradient Descent). Note that all the algorithms have an initial step of finding the optimal rank \( r \) subspace for each group, in which its time complexity is not included here.

error of each group’s data on both subspaces. Thus, when a dimension reduction algorithm is applied to the joint data, the reconstruction loss of some of the groups is degraded (from what they can achieve with their data only), while others are benefiting from joint learning. Here, a fair algorithm is the one that can find a subspace with optimal trade-offs between these degradation and benefits.

An attempt to impose the fairness constraint on learning the optimal subspace for two protected groups has been made recently in \([134, 121, 113]\), where the fair subspace learning is sought by minimizing the maximum deviation of reconstruction error suffered by any protected group (i.e., the difference of per group reconstruction error and joint reconstruction error). Interestingly, it has been shown that at any optimal local solution of the optimization problem associated with learning such a fair subspace, all the groups suffer the same loss. Motivated by this observation, a semi-definite programming relaxation followed by linear programming is proposed to find a fair subspace \([134, 121, 113]\). In addition to the computational inefficiency of algorithms proposed by these works, the generalization of them to multiple group sensitive features is not conspicuous. Furthermore, since all optimal solutions do not incur the same loss for all groups, extra dimensions are needed to ensure that the total loss of the projection remains at most the optimal objective in the original target dimension (in particular, \( k - 1 \) extra dimensions are needed for \( k \) groups in \([134]\) which is further tightened to \( \sqrt{k} \).
in a followup work [113]).

The overarching goal of this chapter is to define a fairness metric for dimension reduction, dubbed as pairwise disparity error, and propose a computationally efficient dimensional reduction algorithm to learn a fair subspace from multiple group sensitive features. Towards this end, we cast the problem of fairness in the PCA dimension reduction algorithm as a multiobjective optimization problem and propose an adaptive gradient descent based approach similar to Algorithm 2 to find the optimal trade-offs with provable convergence rates. Interestingly, the proposed framework is not bounded to any specific notion of fairness metric and can be effortlessly applied to other metrics as well. Moreover, unlike the aforementioned prior works, no extra dimension is needed to ensure the loss suffered by each group matches the optimal fairness loss. The comparison of time complexity of exiting algorithms and current work is summarized in Table 6.1.

**Contributions** The main contributions of this chapter can be summed up as follows:

- We introduce the notion of Pareto fair PCA to ponder conflicting objectives and achieve optimal trade-offs between them. Also, we introduce the notion of pairwise disparity error as a more efficient objective to learn fair subspaces. In addition, we provide conditions, under which a Pareto efficient solutions exist.

- We propose a gradient descent algorithm to efficiently solve the obtained multiobjective optimization problem for dimension reduction problem, and provide theoretical guarantees on its convergence to optimal compromises or a Pareto stationary point of that problem.

- We empirically develop this algorithm and compare it to the state-of-the-art algorithm on two real-world datasets to demonstrate its efficacy that complements our theoretical results.

- We investigate the effect of fair projection on supervised tasks such as classification empirically and show that it can significantly eliminate the unfairness in downstream tasks.

### 6.2 Related Work

The efforts to address fairness in algorithmic decision-making systems have roughly fallen into three different categories. Some scholars believe data itself could be biased,
leading to unfair results; thus, they seek to solve this problem on data level and as a preprocessing step to the main learning task \[41, 48, 83, 24\]. The goal is achieved by either changing the value of sensitive feature or label data or find a subspace, where labels and sensitive features are independent. However, since the main objective of the learning is not involved in this process, the optimal solution for the main objective is not guaranteed.

The second category includes methods that try to impose the fairness criteria after the learning, in order to attain a fair model \[64, 84, 58, 23\]. The third approach, includes methods that try to satisfy fairness constraint during the training procedure, usually by imposing them as a constraint to the main learning objective \[38, 113, 153, 134, 126\]. Some of these approaches treat the fairness problem similar to imbalanced data or rare event prediction \[150, 80, 79, 78, 118\]. While these approaches can achieve the state-of-the-art results in some problems, they still suffer from several issues. Solving a constrained optimization could be a very hard non-convex problem; hence, relaxation is needed to solve the problem that leads to sub-optimal solutions efficiently. Moreover, finding the optimal penalization parameter could be a difficult task, as discussed in \[38\].

Our approach belongs to the third category, yet, it differs from the prevailing trend of formulating the fairness problem as a constrained optimization. We will cast the fairness problem as a multiobjective optimization that can efficiently satisfy fairness objectives as well as the main learning objective and converge to a point with optimal compromises between objectives.

Fairness in dimension reduction algorithms is recently being vetted by \[134\], through which they propose a semi-definite programming and prove that its solution satisfies the proposed notion of fairness. Aside from the inefficiency of solving the SDP, their approach is developed for binary sensitive features and requires one extra dimension to guarantee fairness. To generalize it for multiple group sensitive features with \(k\) groups, they propose to add \(k - 1\) dimensions, which is impractical. The follow-up studies by \[121, 113\] are still in line with the previous one, trying to relax and solve an SDP. We, on the other hand, propose an efficient gradient-based method to solve the aforementioned multiobjective optimization, with the capability of generalizing to multiple group sensitive features smoothly.

Although it has been asserted that fairness problems are multiobjective problems in nature \[88, 101, 113\], as mentioned before, most of the existing works apply different forms of relaxations and approximations to reduce the problem into a scalar-valued optimization problem. In this chapter, we design the fairness problem at hand as a multiobjective optimization and solve it directly. Multiobjective or vector optimization is
a well-studied problem in different domains for many years. The goal in this optimization
is to achieve an optimal trade-off point between different objectives, known as Pareto
efficient, named after Italian economist Vilfredo Pareto. We refer the reader to [112]
and the references therein as a rich resource on multiobjective optimization. We will
elaborate that directly solving the vector-valued problem associated with fair learning is
appealing to reduction based counterparts [43, 109] by being computationally efficient
and providing provable guarantees on the fairness metric.

Beyond achieving fairness in unsupervised tasks such as PCA, the main goal of
fairness in machine learning is to design a fair system as a whole. As it is noted by [42],
these machine learning models in isolation do not necessarily result in a fair system
together and should be considered in composition with each other. Hence, in addition
to what introduced by [39] as compositions in fairness, we advocate for considering the
composition of a stream of machine learning models together. Thus, we should investigate
the effect of imposing fairness constraints on a machine learning model on downstream
tasks using its output. For instance, the goal of defining such a metric for fairness, in
this chapter and other related works, is that having a fair loss in reducing the dimension
would have a fair reduction in the quality of different groups in the new projection;
then, it can have a balanced impact on the quality of a subsequent classifier learned on
that projection. We empirically investigate the effect of this composition and leave its
theoretical understanding to the future work.

### 6.3 Problem Formulation

We start by mathematically defining the problem we ought to solve, and then discuss
what is the notion of fairness in PCA algorithm, which could be quite different from
what is known as fairness measures in supervised learning.

#### 6.3.1 PCA

The main objective of the PCA is to find the best representation of the data $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $n$ data points in $d$-dimensional space, in a lower dimension $r \leq d$ using a
linear transformation, in order to have the minimum reconstruction error. This linear
transformation can be represented by a projection matrix $\mathbf{U} \in \mathbb{R}^{d \times r}$. Thus, the objective
of PCA is to find a projection matrix $\mathbf{U}$ and a recovery matrix $\mathbf{W} \in \mathbb{R}^{r \times d}$ to minimize
this reconstruction error similar to [135]:

$$\arg\min_{U \in \mathbb{R}^{d \times r}, W \in \mathbb{R}^{r \times d}} \|X - XUW\|_F^2$$  \hspace{1cm} (6.1)

It can be proved that in the solution of (6.1), we have \(W = U^\top\), and columns of \(U\) are orthonormal (i.e. \(U^\top U = I_{r \times r}\)). Therefore we can define the reconstruction loss for any PCA projection as follows:

**Definition 3 (Reconstruction Loss).** For any given dataset \(X\) and any projection matrix \(U\), the total reconstruction loss of \(X\) using \(U\) is defined as:

$$\mathcal{L}(U) \triangleq \ell(X; U) = \|X - XUU^\top\|_F^2$$  \hspace{1cm} (6.2)

The optimal subspace with minimum reconstruction loss given \(X\) can be found by solving the above non-convex optimization problem.

### 6.3.2 Fair PCA

In this section, we will formally define the notion of fairness in dimension reduction algorithms such as PCA. As it was discussed before, the problem arises from having different reconstruction losses on different sensitive groups in a dataset. This means that finding an optimal projection matrix \(U^*\) by solving the minimization problem in (6.2), would have different reconstruction loss on data partitions from each sensitive group. However, in this problem, unlike supervised problems previously discussed, we are not able to reach equality between these reconstruction losses for different groups. The reason for that is the subspace for each group’s data is different, and so is the reconstruction error of that data for that projection. We note that while learning a separate (local) subspace for each individual group has the optimal reconstruction error, our focus here is to learn a single global subspace for all groups due to statistical and ethical concerns. In particular, from a statistical standpoint, since the number of training samples for some groups might be small for skewed data sets, joint learning to have more samples to learn a subspace is preferable and more fair. Ethically, as elaborated in [101] and [85], learning separate subspaces (having disparate treatment like in affirmative action) constructs no trade-offs, and it poses several ethical and legal concerns. We note that the case of fairness with decoupled model representations has been investigated by several other works [42, 145, 29].
In order to quantify to what extent each group suffers or benefits from joint subspace learning, we should compare the subspaces learned from each group’s data alone and the one with other groups’ data included. Then, the idea of fairness is to reach a balance between these sacrifices and benefits of different groups. Formally, consider one of the $d$ features of $X$ as a sensitive feature with $k$ different groups, $S = \{s_1, \ldots, s_k\}$. We denote the matrix of each group’s data points as $X_i \in \mathbb{R}^{n_i \times d}$, where $n_i$ is the number of samples belonging to the sensitive group $s_i$. Hence for any projection matrix $U \in \mathbb{R}^{d \times r}$, the reconstruction loss for each group is defined as:

$$L_i(U) \triangleq \ell(X_i; U) = \|X_i - X_iUU^\top\|_F^2, \quad 1 \leq i \leq k. \quad (6.3)$$

Then, if we only use the dataset $X_i$ to learn the projection matrix, we can find the subspace represented by $U_i^*$ that has the optimal reconstruction loss on that dataset, denoted by $L_i(U_i^*)$. Therefore, a fair dimension reduction algorithm is the one that can learn a global projection matrix $U^*$ on all data points with having equal distance between each group’s reconstruction loss on the subspace learned by the whole data with the subspace learned only by its own data. To formally define these fairness criteria, we introduce the notion of disparity error as follows:

**Definition 4 (Disparity Error).** Consider a dataset $X \in \mathbb{R}^{n \times d}$ with $k$ sensitive groups with data matrix $X_i, i = 1, 2, \ldots, k$ representing each sensitive group’s data samples. Let $U_i^* = \arg\min_U L_i(U)$ denote the projection matrix learned only based on $X_i$. Then for any projection matrix $U$ the disparity error for each sensitive group is defined as:

$$\mathcal{E}_i(U) = L_i(U) - L_i(U_i^*), \quad 1 \leq i \leq k. \quad (6.4)$$

This measure shows that how much reconstruction loss we are suffering or enjoying for any global projection matrix $U$, with respect to the reconstruction loss of optimal projection matrix, we can learn locally based on data points $X_i$. Note that calculating the optimal rank $r$ subspace for each group in $L_i(U_i^*)$ has a one-time overhead to the algorithm’s time complexity overall. However, we ignore this overhead, as did other algorithms we are comparing to and leave the joint learning of both local and global subspaces as future work.

Using the Definition 4, we can define a fair PCA algorithm as follows:

**Definition 5 (Fair PCA).** A PCA algorithm with projection matrix $U^*$ is called fair,
if the disparity error among different groups are equal. That is:

$$E_1(U^*) = E_2(U^*) = \ldots = E_k(U^*). \quad (6.5)$$

A subspace $U^*$ that archives the same disparity error for all groups is called a fair subspace.

### 6.4 Pareto Fair Subspace

In this section, we discuss the key challenges in finding a fair subspace using relaxation methods and motivate our formulation of Pareto fair subspace followed by providing conditions sufficient to guarantee the existence of such subspaces.

#### 6.4.1 Relaxation methods and their limitations

A major challenge to find a fair subspace as defined in Definition 5 is to solve the optimization problem that satisfies (6.5), which is essentially a multiple objective optimization problem by nature. To illustrate this and for ease of exposition, let us focus on the binary sensitive feature ($k = 2$), i.e., there are only two groups in the sensitive feature of the data (e.g., male and female), in which the goal of fair PCA is to satisfy:

$$E_1(U^*) = E_2(U^*), \quad (6.6)$$

In [134], it has been shown that by casting the multiobjective optimization problem as a minmax problem of the form

$$\min_{U \in \mathbb{R}^{d \times r}, \text{rank}(U) \leq r} \max \{E_1(U), E_2(U)\}, \quad (6.7)$$

and using an additional dimension for the projection, the optimal solution of minmax problem results in the same loss for both groups (i.e., $E_1(U^*) = E_2(U^*)$).

Motivated by this observation, a semi-definite relaxation to solve the optimization problem is proposed, which is not efficient for a large number of training samples. Also, to achieve their fairness criteria and ensure that the obtained local solution achieves the optimal fairness objective for all groups, the proposed solution requires adding an extra dimension for a binary sensitive feature and $k - 1$ additional dimensions for a $k$-group sensitive feature, which is not reasonable for a large $k$. We note that in [113], the
requirement of extra dimension is improved to \( \lfloor \sqrt{2k + \frac{1}{4}} - \frac{3}{4} \rfloor \), but it still requires extra projection dimensions to satisfy the fairness constraint. Finally, the optimal trade-offs between fairness objectives and total reconstruction loss, in the case of the same target dimension, is not guaranteed, which would lead to a solution that sacrifices too much of the total reconstruction loss to achieve the fairness criteria. In fact, in order to guarantee that the solution of minmax optimization results in a rank \( r \) subspace with optimal fairness objective, one could choose the target dimension to be \( r - s \), where \( s \) is the extra dimensions needed with \( \mathcal{O}(\sqrt{k}) \) followed by a rounding to reach to the target \( r \) dimensional subspace as proposed in [113]; but this remedy hurts the optimal objective value by a multiplicative factor of \( s/r \). This issue becomes more concerning as the number of groups \( k \), and hence \( s \) increases due to the fact that all local optimal solutions might not achieve the same loss for all groups. Consequently, any solution to fair PCA necessitates jointly minimizing the main objective, which is total reconstruction loss in (6.2), and fairness criteria to balance the trade-off between them.

An alternative solution to alleviate aforementioned issues which is explored in [38] is to impose fairness constraints in minimizing the reconstruction loss in (6.1) as additional constraints, i.e.,

\[
\begin{align*}
\min_{U} & \quad \mathcal{L}(U) \\
\text{subject to} & \quad \mathcal{E}_i(U) \leq \epsilon, \; i \in [k].
\end{align*}
\] (6.8)

which reduces the problem into an instance of non-convex constrained optimization problem to find a fair subspace to all sensitive groups. Relaxing the problem of finding the fair subspace as a constrained optimization similar to (6.8), apart from being a hard non-convex problem which is not evident to solve due to presence of non-convex constraints, requires the optimal constraint violation parameter, \( \epsilon \), to be decided heuristically which is a burden on the use and makes the problem even harder. Although using the Lagrangian method we can turn the problem into an unconstrained non-convex optimization problem—a method known as scalarization relaxation for multiobjective optimization counterpart (e.g., please see [43]), deciding the Lagrangian multipliers is as hard as solving the original problem and does not guarantee the optimally of the obtained solution. Also, since the scale of the objectives might be different, it could lead to infeasibility issues in the optimization problem, or some points from the Pareto frontier could not be attained.

To address challenges arising from the above reduction methods, and in order to achieve the optimal trade-offs between objectives and satisfy equality between disparity errors, we aim at directly solving the multiobjective programming [112]. Towards this end, we note that the optimization problem in (6.8) is a relaxation of the following
generalized multiobjective optimization problem:

$$\arg \min_U [\mathcal{L}(U), \psi (\mathcal{E}_1(U)), \ldots, \psi (\mathcal{E}_k(U))]$$  \hspace{1cm} (6.9)$$

where $\psi(\cdot): \mathbb{R} \to \mathbb{R}_+$ is any penalization function, such as $\psi(z) = |z|$, $\psi(z) = \frac{1}{2}z^2$, or $\psi(z) = e^{-z}$, however, for convergence analysis we will stick to squared or exponential penalization due to their smoothness. We will define the optimization problem in more detail in the next section and then will introduce an adaptive gradient descent approach to solve it.

### 6.4.2 Pareto fair subspace

To characterize the solutions obtained by directly solving the multiobjective optimization problem in (6.9), we have to compare the objective vector of different solutions with each other, analogous to what we do in a scalar or single-objective optimization problem. Recalling the definition of dominance in Definition 1 from Chapter 4, for an objective vector $f(U) = [f_1(U), \ldots, f_m(U)]^\top$, we can say solution $U_1$ is better than $U_2$, if its corresponding objective vector dominates the other solution’s objective vector. That is for these two solutions we can write $f(U_1) \prec_p f(U_2)$.

The definition of dominance implies that when a solution cannot be dominated by any other solution in the search space, we cannot find any direction, to move to, from this solution without at least hurting one objective in the objective vector. Using this, now, we can define our notion of Pareto fair subspace as follows:

**Definition 6 (Pareto Fair Subspace).** Let $f(U) = [\mathcal{L}(U), f_1(U), \ldots, f_{m-1}(U)]^\top$ denote a vector-valued objective function with $m$ objectives in the Fair PCA problem. Then, consider a set of fairness trade-off objectives $f_i(U), i \in [m-1]$, (e.g. $\psi(\mathcal{E}_i(U))$ as in (6.9)) that ought to be minimized in addition to the main objective, $\mathcal{L}(U)$. The solution $U^*$ is called Pareto fair subspace, if it is not dominated by any other feasible solution.

Similar to Pareto efficient points, the Pareto fair subspace is not unique, and the set of Pareto efficient solutions is called Pareto frontier [112]. Thereupon, the ultimate goal of a fair PCA reduces to finding a Pareto efficient solution via solving the problem (6.9).
6.4.3 Proof of existence

The following theorem establishes the conditions under which the set of Pareto efficient solutions exists and is non-empty. We emphasize that compared to methods that optimize Lagrangian function or other scalarization approaches, we aim at finding this Pareto fair frontier completely without any prior information such as weight for each objective.

**Theorem 5 (Existence).** Consider the vector-valued optimization problem in (6.9). If the individual functions are convex and bounded, then the set of Pareto efficient solutions is non-empty.

**Proof.** The proof is deferred to Section 6.8.1.

To guarantee the existence of a Pareto efficient solution, in Section 6.5, we convexify the objectives by properly regularizing them. Thereafter, we propose an efficient gradient-based algorithm to find a subspace that is a Pareto stationary point of the fair PCA problem.

Although solving the optimization problem in (6.9) results in an efficient trade-off between different objectives, this does not reflect on balanced disparity errors among different groups, which is the ultimate goal of the fair PCA problem. As been explained by [134], this issue would be exacerbated in problems with $k > 2$, that having a balanced disparity error among all groups is not always possible due to the fact that all optimal solutions will not incur the same loss for all groups. To alleviate this issue and ensure that the loss of each group remains at most the optimal fairness objective in the original target dimension $r$, we introduce the notion of pairwise disparity error, that would address this issue.

**Definition 7 (Pairwise Disparity Error).** Consider the disparity errors for any projection matrix $U$ and sensitive groups of $i$ and $j$ among $k$ different groups, then the pairwise disparity error between these two groups is defined as:

$$\Delta_{i,j} = \mathcal{E}_i (U) - \mathcal{E}_j (U), \quad i, j \in [k], i \neq j. \quad (6.10)$$

Thus, the optimization in (6.9) becomes:

$$\arg \min_U [\mathcal{L}(U), \psi(\Delta_{1,2}(U)), \ldots, \psi(\Delta_{k-1,k}(U))], \quad (6.11)$$
where we have \( \binom{n}{k} \) objectives in addition to the main objective. We will show the efficacy of pairwise disparity error over single disparity error in practice in Section 6.6.

## 6.5 Adaptive Optimization

In this section, we will develop a gradient descent (GD) based algorithm, on the ground of the proposed PDO in Algorithm 2, to solve the optimization problems in (6.9) or (6.11). To lay the groundwork for this algorithm, we review how to solve the original PCA problem using gradient descent, and then we propose our proposed algorithm to solve the aforementioned multiobjective problem.

### 6.5.1 Gradient descent for PCA

To solve the PCA problem using the gradient descent approach, we need to iteratively update the projection matrix \( U \), based on the gradient of the total reconstruction loss with respect to it. Expanding the total reconstruction loss in (6.2) and removing the constant terms that will not affect the optimization, following [135], we can write the optimization problem:

\[
\arg\min_{U \in \mathbb{R}^{d \times r}, U^\top U = I} -\text{tr} \left( U^\top X^\top X U \right),
\]

(6.12)

Using (6.12), we can calculate the gradient of the total reconstruction loss with respect to \( U \) as follows:

\[
G(U) = \frac{\partial \mathcal{L}(U)}{\partial U} = -2X^\top X U.
\]

(6.13)

The projection can be learned using the gradient descent by iteratively updating an initial solution by:

\[
U_{t+1} = \Pi_{P_r} \left( U_t - \eta_t G(U_t) \right),
\]

(6.14)

where \( \eta_t \) is the learning rate and \( \Pi_{P_r} (\cdot) \) is the projection operator onto the Stiefel manifold \( P_r = \left\{ U \in \mathbb{R}^{d \times r} \mid U^\top U = I_r \right\} \).

For a single-objective optimization like normal PCA, at each iteration, we take a step toward the negative of the gradient at that point. However, when we are dealing with multiple objectives, the key question is what would be the best direction at each iteration to take, in order to decrease all the objectives. We answer this question in the next section by proposing an optimization problem to find such a descent direction.
Algorithm 4: Pareto Fair PCA

**Input:** \( r, X = X_1 \cup X_1 \cup \ldots \cup X_k, U_0 \in \mathbb{R}^{d \times r}, T \)

Find the optimal \( r \)-rank subspace for each group, \( U^* = \{ U_1^*, \ldots, U_k^* \} \) \( \text{ // e.g., using SVD or SGD } [136] \)

**function** ParetoFairPCA\((r, X, U^*, U_0, T)\):

Form the objective vector \( f(U) \) using \( U^* \) and \( X \) from (6.9) or (6.11)

for \( t = 1, \ldots, T \) do

Calculate the gradient of each objective, \( G_i^{(t)} \)

Find the descent direction \( D^{(t)} \) using (6.16)

if \( D_t = 0 \) then

| return \( U_t \) \( \quad \text{ // Pareto stationary point} \)

end

Find minimum \( p \in \mathbb{N} \) such that for \( \eta_t = \frac{1}{2^p} \): \( \quad \text{ // Backtracking line search (optional) } \)

\[
\begin{align*}
    f_i(U_t + \eta_t D_t) &\leq f_i(U_t) + \beta \eta_t \text{tr} \left( D_t^\top G_i^{(t)} \right), \ i \in [m].
\end{align*}
\]

(6.15)

\[
U_{t+1} = \Pi_{P_r} (U_t + \eta_t D_t)
\]

end

return \( U_{T+1} \)

---

6.5.2 Pareto fair PCA

In order to efficiently solve the multiobjective optimization problem in (6.9) or (6.11), we propose a gradient descent approach, that can guarantee convergence to a Pareto stationary point. For the ease of exposition, we consider the following general multiobjective problem with \( m \) objectives:

\[
f(U) = [f_1(U), \ldots, f_m(U)]
\]

\[
D_t = \arg \min_{D \in \mathbb{R}^{d \times r}} \left\{ \max_{i=1, \ldots, m} \text{tr} \left( D^\top G_i^{(t)} \right) + \frac{1}{2} \|D\|_F^2 \right\}. \quad (6.16)
\]

We note that for a single objective case, that is \( m = 1 \), the solution of above minimax is the opposite of the gradient, i.e. \( D_t = -G_1^{(t)} \). Using the KKT optimally conditions, it is easy to show that the dual problem becomes a quadratic programming and can be efficiently solved to identify a descent direction \( D_t \), for which all the objectives are non-increasing. The following lemma states this characteristic of the descent direction:

**Lemma 5** (Descent Direction). **The solution found in the optimization problem (6.16) has one of the following two conditions.** Either \( D_t = 0 \), which means the point \( U_t \) is a...
Pareto stationary point, or $D_t$ is a descent direction to all objectives, that is:

$$\text{tr} \left( D_t^\top G_i^{(t)} \right) \leq 0, \ \forall \ 1 \leq i \leq m$$

(6.17)

Then, the obtained descent direction is in the form of $D_t = -\sum_{i=1}^{m} \lambda_i^{(t)} G_i^{(t)}$, where $\sum_{i=1}^{m} \lambda_i^{(t)} = 1$ and $\lambda_i^{(t)} \geq 0$ for $1 \leq i \leq m$.

**Proof.** The proof is provided in Section 6.8.2. □

As elaborated in the proof in Appendix 6.8.2, the lemma implies that the descent direction is the minimum norm matrix in the convex hull of the gradients of all objectives and is the non-increasing direction with respect to each objective. Understanding this, the following corollary is palpable:

**Corollary 1.** The first order Pareto stationary point holds for a solution $U$ when the mentioned minimum norm is zero, i.e., there is no descent direction that is non-increasing for all objectives. In other words, there exists a $\lambda \in \Delta_m$ such that $D = -\sum_{i=1}^{m} \lambda_i G_i = 0$ where $G_i = \nabla f_i(U)$.

Having a descent direction at hand, we can use it to decrease all the objectives in every iteration, similar to the procedure defined in Algorithm 4. Based on the first-order optimality condition of this problem, we know that at a Pareto efficient solution, the direction found in (6.16) should be 0, meaning, that it cannot further improve any objective without hurting others. Equipped with this descent direction and first-order optimality condition, we can iteratively update the initial solution in the direction of the descent direction, until it converges to a Pareto stationary point.

**Remark 6.** One crucial step before finding the descent direction is to balance out the scale of different gradients. Since they are calculated based on very different and possibly contradictory objective functions, their Frobenius norm would vary a lot; hence, by a normalization step, we can avoid the dominance of the descent direction by some gradients with high Frobenius norm.

Since the disparity errors, as well as the main PCA objective, are weakly convex functions, following Theorem 5, to guarantee the existence of Pareto efficient subspace, we add a regularization term to each objective to make them convex functions—which we also stabilize the solutions and guarantee convergence. As a result, the optimization
in (6.11) becomes:

\[
\begin{aligned}
\arg\min U & \left\{ \mathcal{L}(U) + \alpha \|U\|_F^2 \\
& \psi(\Delta_{1,2}(U)) + \alpha \|U\|_F^2 \\
& \vdots \\
& \psi(\Delta_{k-1,k}(U)) + \alpha \|U\|_F^2 \\
\right\} ,
\end{aligned}
\]

(6.18)

where \( \alpha \) is the regularization parameter to make the Hessian matrices of objectives positive semi-definite and needs to be decided based on the maximum eigen-gap between covariance matrices of each pair of sensitive groups. Having \( k \) different groups, each with data matrix of \( X_i, i \in [k] \), we set \( \gamma = \max_{i,j \in [k]} \gamma_{id}(X_i^\top X_i) - \gamma_1(X_j^\top X_j) \) to denote the maximum eigen-gap. Then, we should have \( \alpha \geq \gamma \). We now turn to prove the convergence rate of Algorithm 4 for convex objectives, as stated in the following theorem.

**Theorem 6 (Convex Objectives).** Let \( f = [f_1(U), \ldots, f_m(U)] \) be convex component-wise Lipchitz continuous with constants \( L_1, L_2, \ldots, L_m \). Then, for the sequence of the solutions \( U_1, \ldots, U_T \) generated iteratively by Algorithm 4, and the sequence of \( \hat{\lambda}_{(1)}, \ldots, \hat{\lambda}_{(T)} \) generated by (6.36) during \( T \) iterations, by setting \( \eta = \frac{R}{L \sqrt{T}} \) and \( \beta = \sqrt{T}/R \), we have:

\[
\sum_{i=1}^m \lambda_i \left( f_i(U_T) - f_i(U^*) \right) \leq \frac{RL}{2\sqrt{T}},
\]

(6.19)

where \( R^2 = \|U_1 - U^*\|_F^2 \), \( L = \max_{i=1,\ldots,m} L_i \), \( \lambda_i = \frac{1}{T} \sum_{t=1}^T \hat{\lambda}_{i(t)} \), and \( U^* \) is a Pareto efficient solution.

**Proof.** The detailed proof is deferred to Section 6.8.3.

Theorem 6 indicates that, using the Pareto descent direction, we can achieve an \( \epsilon \)-accurate Pareto efficient solution with taking \( \mathcal{O}\left(\frac{1}{\epsilon^2}\right) \) gradient descent steps. Using (6.19), we can bound the average deviation of each objective from its respective value in the Pareto efficient solution of \( U^* \).

We note that when the regularization is not added to convexify the main objective, we have to deal with non-convex objectives in the optimization problem. In the following theorem, we investigate the convergence of Algorithm 4 for non-convex objectives that guarantees the gradient vanishes over iterations.

**Theorem 7 (Nonconvex Objectives).** Let \( f(U) = [f_1(U), \ldots, f_m(U)] \) be the multiobjective function to be minimized to find a fair subspace with respect to \( k \) sensitive groups. Let \( U_1, U_2, \ldots, U_T \) be the sequence of solutions generated by Algorithm 4 updated using
descent directions $D_1, D_2, \ldots, D_T$. Then, if we choose the regularization parameter as $\alpha \geq \gamma$, we have the following:

$$
\min_{t=1,2,\ldots,T} \|D_t\|_F \leq \sqrt{\frac{M_u - M_l}{CT}},
$$

(6.20)

where $M_l$ is a lower bound for the values of all objective functions, $M_u$ is the maximum of the values of all functions at initial point, and $C$ is a constant depending on the smoothness of objectives.

Proof. The proof can be found in Section 6.8.4.

An immediate consequence of the above theorem is that the gradient of Pareto descent directions vanishes and converges to zero and thereby the solutions generated by the algorithm converges to a stationary fair subspace. In particular, only $O\left(\frac{1}{\epsilon^2}\right)$ iterations are required to obtain an $\epsilon$-close fair subspace. The analysis of Theorem 7 follows the standard analysis of gradient descent for non-convex smooth optimization where the obtained bound matches the known achievable convergence rate for the norm of the gradients. We want to sketch another alternative method that results in the same rate with careful analysis. Specifically, observe that the descent direction can be considered as an inexact gradient from the viewpoint of individual functions with perturbation $D_t - G_i^{(t)}$. Noting that $\text{tr} \left(D_t^\top G_j^{(t)}\right) \leq -\|D_t\|_F^2$ as shown in the proof of Lemma 5 and following the standard analysis of convergence of non-convex functions, we can show that norm of descent directions vanishes as algorithm proceeds, thereby the proposed algorithm can find a stationary point. However, the obtained solution is not guaranteed to be an optimal Pareto due to the non-convexity of the objectives and might be a saddle point.

### 6.5.3 Comparison with other approaches

As it was discussed, one approach to solve a multiobjective optimization is to make it constrained optimization, in which we keep the main objective and change all other objectives to inequality constraints with parameters $\epsilon$. Hence, constrained optimization is a relaxation of multiobjective optimization, where finding the best constraint parameter ($\epsilon$) for each constraint could be very challenging as discussed in [38]. It also lacks theoretical guarantees due to the non-convex nature of constraints. Lagrangian method of multipliers is equivalent to constraint optimization problems, but not exactly to multiobjective counterpart. To see this, we note that by applying GD to Lagrangian
function, the contribution of the gradient of each individual function, \( G^{(t)}_i \), is weighted by its Lagrangian multiplier, while in our case the weights are adaptively learned by finding a Pareto decent direction. We note that while [113] improves the requirement of extra dimensions over [134], it still needs \( \lfloor \sqrt{2k + \frac{1}{4} - \frac{3}{2}} \rfloor \) extra dimensions for a \( k \)-group sensitive feature and has to solve an SDP which has the time complexity of \( \mathcal{O}(d^{0.5} \log \left( \frac{1}{\epsilon} \right)) \) or \( \mathcal{O}(d^8/\epsilon^2) \) with multiplicative weight update. On the other hand, our method enjoys the efficiency of GD with an overhead to solve the quadratic problem over the simplex for finding the descent direction. Also, at each step, we need to project the solution to find the orthonormal bases for the updated solution, which could be done using SVD with an overhead of \( \mathcal{O}(r^2d) \) or more efficiently using variance-reduced SGD [136]. Using vanilla SVD for per-iteration projection brings the overall time complexity of the proposed algorithm to \( \mathcal{O}(r^2d/\epsilon^2) \). We note that the convex formulation in [121] also requires solving an SDP programming (e.g., ellipsoid method to interior point method), which suffers from high computational cost as well.

This is for the first time that we are solving the exact multiobjective problem, rather than its min-max relaxation using SDP in a fairness problem. [113] is suggesting that for \( k = 2 \) their approximation is exact, meaning their algorithm will find the fair representation in the exact \( r \) dimension they aim to reach. However, in practice, even for \( k = 2 \), we can show that our algorithm can achieve a smaller disparity error, as shown in Figure 6.2, which indicates that pairwise disparity error can achieve a better subspace in terms of fairness. For \( k > 2 \), they are still solving an inefficient SDP problem to exact same problem we are proposing. Hence, the novelty of our approach lies in solving this problem using gradient descent and ensuring to reach a Pareto stationary point, which even does not require extra dimensions to satisfy fairness. This setting and its proposed gradient descent algorithm to solve it can be applied to other unsupervised and supervised fairness problems. Thus, it could open up new perspectives on all other fairness problems in learning tasks, by advocating optimal trade-offs between main learning objectives and fairness criteria using Pareto efficiency.

### 6.6 Experiment

In this section, we empirically examine the introduced algorithm for fair PCA with the Adult dataset and the Credit dataset, similar to the Chapter 5. The Adult dataset consists of census data to predict whether the income of a person exceeds 50K per year or not. The Credit dataset contains clients’ credit history information to predict whether
they would default in the future or not. For PCA, we will omit the label data and work with the rest of it, which contains 14 features for the Adult dataset, including gender and race, which we consider as sensitive features in this dataset. In the Credit dataset, we will have 23 features, including sensitive features of sex and marriage. The gender feature in the Adult dataset and sex in the Credit dataset are binary features with two values, namely, Male and Female. Race from the Adult dataset, on the other side, is a multiple group feature, with 5 different groups, including White, Asian-Pac-Islander, Amer-Indian-Eskimo, Black, and Other. Marriage in the Credit dataset is also a multiple group feature with 3 groups of Single, Married, and Other.

For the Adult dataset, we use the training dataset, which has 32,561 number of samples, among which 10,548 belongs to the Female group and 22,013 to the Male group. The distribution of samples among race groups are as follows: Black 30,47, White 27,994, Asian-Pac-Islander 312, Amer-Indian-Eskimo 962, and Other 246. In the Credit dataset, we have 30,000 training samples, out of which there are 18,112 Female and 11,888 Male samples. The distribution of the Marriage feature is 13,659 married, 15,964 single, and 323 other samples. We first apply the fair PCA method to binary sensitive feature, in which we set the learning rate to $1/\sqrt{t}$, where $t$ is the iteration number. This condition on the learning rate satisfies the maximum decrease condition by backtracking line search in (6.15).

### 6.6.1 Binary sensitive feature

In the Adult dataset, we observed that the Female group is benefiting from normal PCA on the whole dataset, while the Male group is sacrificing its reconstruction error. Hence, by applying the Fair PCA algorithm, we can perfectly decrease these trade-offs, while suffering an insignificant loss to the total reconstruction error, compared to normal PCA. The results are depicted in Figure 6.1, where the trade-offs and how Fair PCA is addressing them is noticeable. To compare the introduced Pareto fair PCA with algorithms using SDP in [134] and [113], we will use the average disparity errors across sensitive groups in both Adult and Credit datasets. Figure 6.2 shows the average disparity errors of Pareto fair PCA with single and pairwise disparity error objectives, SDP fair PCA, and normal PCA on binary features (gender and sex) of Adult and Credit datasets. First, it reveals that there is a huge gap between normal PCA and fair PCA algorithms in terms of disparity errors, which is indicating that these algorithms are decreasing this disparity error. Second, it shows the superiority of Pareto fair PCA over SDP relaxation methods in both datasets (especially with pairwise objectives), where Pareto fair PCA
Figure 6.1: Applying normal PCA and fair PCA to the Adult dataset with gender as its sensitive feature. The first two figures show the reconstruction error of normal PCA (trained on all data) applied to each group, fair PCA (trained on all data) applied to each group, and normal PCA trained on the data of each group individually. The last figure reveals the difference between normal and fair PCA reconstruction loss on all data, which is very tiny and negligible.

Figure 6.2: Comparing Pareto fair PCA algorithm introduced in this chapter (pairwise and single disparity error) with fair PCA algorithms using SDP relaxation introduced in [134] and [113]. The experiment is on binary features of Adult and Credit datasets (gender and sex). The average disparity error of algorithms on the Adult dataset clearly shows the superiority of the Pareto fair PCA with pairwise disparity error objectives and then the single disparity error objectives. In the Credit dataset, Pareto fair PCA with pairwise objectives has a slightly better performance with respect to two other methods.

has a smaller average disparity error close to zero. Also, to show what is the exact price of fairness that each algorithm pays, we show the total reconstruction loss of our algorithm and other fair PCA methods with respect to normal PCA in Figure 6.3. It can be noted that the Pareto fair PCA incurs a slight degradation in total reconstruction loss in exchange for fairness, while this price is much higher in the other state-of-the-art algorithms.
Adult and Credit datasets (gender and sex). From both figures, it can be inferred that we show the disparity error of different groups in Figure 6.5, which reveals that fair The proposed Algorithm 4, can efficiently generalize to the multiple group sensitive feature. The results indicate that even in a dataset with a multiple group sensitive feature, the increase in the reconstruction loss of fair PCA compared to the normal PCA is slim, while the gap between their disparity errors is huge. This means that normal PCA is unfairly treating features, by adding pairwise disparity errors of each pair of groups to the objective vector and minimize the overall vector to reach a Pareto efficient or stationary point. However, adding more objectives, introduces more trade-offs, makes the optimization over all objectives more difficult.

First, we start with the Adult dataset with race as the sensitive feature. In this dataset, race has 5 categories, makes it a multiple group sensitive feature. The reconstruction error of the Pareto fair PCA and normal PCA is shown in Figure 6.4, where the trade-offs between benefits and sacrifices of different groups are clearly noticeable. The fair PCA algorithm can superbly decrease these trade-offs for all but one group, with a negligible increase in overall reconstruction loss. Following the same step as in the binary case, we show the disparity error of different groups in Figure 6.5, which reveals that fair PCA clearly outperforms normal PCA in most of the groups. Figure 6.6 depicts the reconstruction loss and average disparity error of fair and normal PCA. The results indicate that even in a dataset with a multiple group sensitive feature, the increase in the reconstruction loss of fair PCA compared to the normal PCA is slim, while the gap between their disparity errors is huge. This means that normal PCA is unfairly treating

6.6.2 Multiple group sensitive feature

The proposed Algorithm 4, can efficiently generalize to the multiple group sensitive features, by adding pairwise disparity errors of each pair of groups to the objective vector and minimize the overall vector to reach a Pareto efficient or stationary point. However, adding more objectives, introduces more trade-offs, makes the optimization over all objectives more difficult.

First, we start with the Adult dataset with race as the sensitive feature. In this dataset, race has 5 categories, makes it a multiple group sensitive feature. The reconstruction error of the Pareto fair PCA and normal PCA is shown in Figure 6.4, where the trade-offs between benefits and sacrifices of different groups are clearly noticeable. The fair PCA algorithm can superbly decrease these trade-offs for all but one group, with a negligible increase in overall reconstruction loss. Following the same step as in the binary case, we show the disparity error of different groups in Figure 6.5, which reveals that fair PCA clearly outperforms normal PCA in most of the groups. Figure 6.6 depicts the reconstruction loss and average disparity error of fair and normal PCA. The results indicate that even in a dataset with a multiple group sensitive feature, the increase in the reconstruction loss of fair PCA compared to the normal PCA is slim, while the gap between their disparity errors is huge. This means that normal PCA is unfairly treating
As for the Credit dataset, we also test it on its multiple group sensitive feature, marriage, which has 3 different groups. The result of reconstruction error on Pareto fair PCA, normal PCA and normal PCA on each group’s data individually is depicted in Figure 6.7, where it is clear that fair PCA is very close to each group’s PCA (except for the “other” group, because the number of samples in that group is too low), while its reconstruction error is very close to that of normal PCA. Also, the disparity error and average disparity error of fair PCA versus normal PCA is shown in Figure 6.8, where the superiority of fair PCA is noticeable.

6.6.3 Fairness in composition

Most of the time, when we use a dimension reduction algorithm, it is accompanied by some downstream tasks such as classifiers. Hence, it is important to investigate the effects of our fairness dimension reduction algorithm on those downstream tasks. Here, we empirically examine this effect on a simple classifier. To that end, we use both the
Figure 6.5: Disparity error of normal and fair PCA trained on the Adult dataset with “race” as its sensitive feature. Each plot depicts the disparity errors of different groups with normal and fair PCA.

Figure 6.6: Applying normal and fair PCA on the Adult dataset with “race” as its multiple group sensitive feature. Left shows the difference between reconstruction loss of fair and normal PCA, which is infinitesimal. On the other hand, the right plot shows their difference in terms of average disparity error, which is huge and demonstrating the efficacy of fair PCA in addressing fairness even in multiple group sensitive feature cases.

Adult and Credit datasets and first reduce the dimension of their feature space to 10, and then use the new projection to learn a standard linear SVM model. One standard fairness measure in the supervised domain is called Equality of Opportunity [64], where
Figure 6.7: Applying normal and fair PCA on the Credit dataset with “Marriage” as its sensitive feature. Each plot shows the reconstruction error of the normal PCA (trained on the whole data) applied to each group’s data, fair PCA (trained on the whole data) applied to each group’s data, and normal PCA trained on each group’s data individually. The last figure shows the reconstruction error of normal PCA and fair PCA on this dataset with multiple group sensitive feature of marriage.

the goal is to ensure that the true positive rate among different sensitive features does not differ significantly. For a binary sensitive group [38] introduced a measure called difference of equality of opportunity, which is $DEO = |TP_1 - TP_2|$ with $TP_i$ representing true positive rate of the $i$th group in a sensitive feature. This measure shows the gap between the two groups’ true positive rates. As can be inferred from Figure 6.9, applying Pareto fair PCA can boost fairness of the downstream model and dramatically drop the gap between two groups’ true positive rates (DEO) with respect to the normal PCA.

### 6.7 Final Remarks

In this chapter, we cast the fairness problem in dimension reduction algorithms such as PCA as a multiobjective programming. Unlike supervised learning, there is not a clear definition of fairness in unsupervised learning tasks. Thus, we use the notion of balancing between sacrifices and benefits each sensitive group makes or enjoys to define
Figure 6.8: Disparity error of normal and fair PCA trained on the Credit dataset with “Marriage” as its sensitive feature. Each plot depicts the disparity errors of different groups with normal and fair PCA. The last figure shows the average of disparity errors across groups.

a fairness metric for this problem. These sacrifices or benefits are the consequence of finding the optimal subspace over the whole data rather than using only each protected group’s data. Hence, the notion of fairness is to have an equal contribution from each group to the overall reconstruction loss with respect to the reconstruction loss they have on the subspace learned by their own data. This introduces a trade-off between these contributions and overall reconstruction loss. We propose an efficient multiobjective optimization procedure that can guarantee the convergence to a Pareto stationary point, which has an efficient trade-off between these objectives.

This proposal also introduces some interesting problems worthy of future investigations. First, the generalization of the proposed disparity error and pairwise disparity error as fairness metrics in other dimension reduction algorithms and, also, other unsupervised learning tasks. Moreover, it is interesting to investigate the stochastic version of the proposed algorithm and its convergence analysis since finding a descent direction where gradients are noisy might be a challenging task. Also, as noted before, the existing methods, including the one proposed in the present work, require learning a local optimal projection subspace for each group before learning the global fair subspace. One
Figure 6.9: The effect of Pareto fair PCA on a downstream SVM classification task. The first row is using the Adult dataset with gender as its sensitive feature, and the second row is using the Credit dataset with sex as its sensitive feature. In both datasets, we reduce their feature space dimension to 10 once using normal PCA and once using Pareto fair PCA. Then use the new representation to learn a linear SVM. Column (a) is the accuracy among different groups, (b) is the true positive rate, and (c) is the DEO introduced by [38] (the lower is better), all on the test dataset. It clearly can be noted that applying Pareto fair PCA can reduce the gap between true positive rates of different groups and enhance the fairness of downstream models.

interesting direction is to extend these works to efficiently learn all subspaces together while preserving the fairness of the global subspace. Finally, a thorough theoretical investigation of the composition effects of the proposed fairness measure on downstream tasks such as classification is an interesting open problem.

6.8 Missing Proofs

In this section, the missing proofs from the main body of this chapter is presented and discussed.
6.8.1 Proof of Theorem 5

Proof. Consider the following constrained optimization problem:

$$\sup_{\epsilon_i \in [m]} \sum_{i \in [m]} \epsilon_i$$
subject to $$f_i(U) + \epsilon_i = f_i(\tilde{U}), \ i \in [m],$$
$$\epsilon_i \geq 0, \ i \in [m],$$ (6.21)

where $\tilde{U}$ is any feasible subspace. By assuming that $f_i(.)$ is convex for $i \in [m]$, if there is no finite maximum value for this optimization, then the set of proper Pareto optimal solutions is empty. The main immediate implication of this theorem is that if the objectives are bounded, then a Pareto optimal solution exists for this optimization problem. More specifically, if the solution of this optimization is the objective value of zero, then the $\tilde{U}$ is a Pareto optimal solution. To prove this theorem, we consider $U^*$ to be a proper Pareto optimal solution to the problem (6.21), then there exists a vector $\lambda \in \mathbb{R}_+^m$, such that the point $U^*$ is a Pareto optimal solution to the problem:

$$\arg \min_U \sum_{i \in [m]} \lambda_i f_i(U)$$ (6.22)

Then, from the Pareto optimality we have for every feasible $U$:

$$\sum_{i \in [m]} \lambda_i [f_i(U) - f_i(U^*)] \geq 0$$ (6.23)

By setting $U = \tilde{U}$, we can write:

$$\sum_{i \in [m]} \lambda_i \left[ f_i(\tilde{U}) - f_i(U^*) \right] = M^\dagger \geq 0$$ (6.24)

Also, from the optimization problem (6.21) since there is not a finite maximum objective value available, for every $\tilde{M} \geq 0$ we can find a $\tilde{U}$ such that:

$$\sum_{i \in [m]} \left[ f_i(\tilde{U}) - f_i(U) \right] \geq \tilde{M}$$ (6.25)
Then, if we set $\lambda_{\min} = \min \{\lambda_1, \ldots, \lambda_m\}$, we have:

$$
\lambda_{\min} \tilde{M} \leq \lambda_{\min} \sum_{i \in [m]} \left[ f_i (\tilde{U}) - f_i (\bar{U}) \right]
= \sum_{i \in [m]} \lambda_{\min} \left[ f_i (\tilde{U}) - f_i (\bar{U}) \right]
\leq \sum_{i \in [m]} \lambda_i \left[ f_i (\tilde{U}) - f_i (\bar{U}) \right]
$$

(6.26)

If the $\tilde{U}$ is chosen to satisfy $\lambda_{\min} \tilde{M} = M^\dagger$, then we have:

$$
\sum_{i \in [m]} \lambda_i \left[ f_i (\tilde{U}) - f_i (U^*) \right] \leq \sum_{i \in [m]} \lambda_i \left[ f_i (\tilde{U}) - f_i (\bar{U}) \right]
$$

$$
\sum_{i \in [m]} \lambda_i f_i (\tilde{U}) \leq \sum_{i \in [m]} \lambda_i f_i (U^*)
$$

(6.27)

which contradicts the assumption of Pareto optimality of $U^*$, and hence, Pareto optimal set is empty.

\[\square\]

6.8.2 Proof of Lemma 5

Proof. The proof is straightforward and directly follows from KKT optimally conditions for problem (6.28), however, we show the derivation here for completeness.

First, we note that the minmax optimization problem introduced in (6.17) to find the descent direction $D_t$, can be rewritten as the following equivalent constrained optimization problem:

$$
(D_t, \epsilon_t) = \arg \min_{D \in \mathbb{R}^{d \times r}, \epsilon \in \mathbb{R}^+} \epsilon + \frac{1}{2} \|D\|_F^2,
\text{s.t. } \text{tr} \left( D^\top G_i^{(t)} \right) \leq \epsilon, \quad \forall \ 1 \leq i \leq m.
$$

(6.28)

Forming the Lagrangian of the constrained problem as follows

$$
\mathcal{L} (D, \epsilon; \lambda_i) = \frac{1}{2} \|D\|_F^2 + \epsilon + \sum_{i=1}^{m} \lambda_i \left( \text{tr} \left( D^\top G_i^{(t)} \right) - \epsilon \right),
$$

(6.29)

and writing the KKT conditions gives:

$$
\frac{\partial \mathcal{L}}{\partial D} = D + \sum_{i=1}^{m} \lambda_i^{(t)} G_i^{(t)} = 0
$$

(6.30)

$$
\frac{\partial \mathcal{L}}{\partial \epsilon} = 1 - \sum_{i=1}^{m} \lambda_i = 0
$$

(6.31)
\[
\frac{\partial \mathcal{L}}{\partial \lambda_i} = \text{tr} \left( \mathbf{D}^\top \mathbf{G}_i^{(t)} \right) - \epsilon = 0 \quad \forall 1 \leq i \leq m \tag{6.32}
\]
\[
\lambda_i \left( \text{tr} \left( \mathbf{D}^\top \mathbf{G}_i^{(t)} \right) - \epsilon \right) = 0 \quad \forall 1 \leq i \leq m \tag{6.33}
\]
\[
\lambda_i \geq 0 \quad \forall 1 \leq i \leq m \tag{6.34}
\]

From (6.30) we have the following that holds for the descent direction:
\[
\mathbf{D}_t = - \sum_{i=1}^{m} \lambda_i^{(t)} \mathbf{G}_i^{(t)}, \tag{6.35}
\]
where \( \lambda^{(t)} = [\lambda_1^{(t)}, \ldots, \lambda_m^{(t)}]^\top \) belongs to \( \Delta_m \) – the \( m \)-dimensional simplex. By plugging these conditions back to the main problem, the dual problem can be simplified as:
\[
\hat{\lambda}^{(t)} = \arg \min_{\lambda \in \Delta_m} \frac{1}{2} \left\| \sum_{i=1}^{m} \lambda_i \mathbf{G}_i^{(t)} \right\|_F^2. \tag{6.36}
\]

By solving the dual problem, which is a quadratic programming and using (6.35) we can find the descent direction from optimal dual variables.

Next, we need to show that the obtained direction is either \( \mathbf{0} \) or a descent direction to all objectives. If the point \( \mathbf{U}_t \) is a Pareto stationary point, then it means that we cannot find a direction that can decrease all the objectives, without increasing one. Hence, there is no such a \( \mathbf{D} \) that \( \text{tr} \left( \mathbf{D}^\top \mathbf{G}_i^{(t)} \right) \leq 0 \) for all \( 1 \leq i \leq m \), unless \( \mathbf{D} = \mathbf{0} \). For points that are not Pareto stationary, consider the following quadratic optimization for every \( 1 \leq j \leq m \):
\[
\arg \min_{\beta \in [0, 1]} \frac{1}{2} \left\| (1 - \beta) \mathbf{G}_j^{(t)} - \beta \mathbf{D}_t \right\|_F^2. \tag{6.37}
\]

We can see that this optimization problem is equivalent to the optimization problem in (6.36), with \( \lambda_i = \beta \hat{\lambda}_i \) for \( 1 \leq i \leq m, \ i \neq j \), and \( \lambda_j = 1 - \beta(1 - \hat{\lambda}_j) \). This means that the optimum of the quadratic optimization in (6.37) happens at \( \beta = 1 \). Then by using the first order optimally condition at optimum point we get:
\[
2 \left\| \mathbf{G}_j^{(t)} + \mathbf{D}_t \right\|_F^2 - 2 \text{tr} \left( \left( \mathbf{G}_j^{(t)} \right)^\top \left( \mathbf{G}_j^{(t)} + \mathbf{D}_t \right) \right) \leq 0 \tag{6.38}
\]
\[
2 \left\| \mathbf{G}_j^{(t)} + \mathbf{D}_t \right\|_F^2 - 2 \text{tr} \left( \left( \mathbf{G}_j^{(t)} + \mathbf{D}_t - \mathbf{D}_t \right)^\top \left( \mathbf{G}_j^{(t)} + \mathbf{D}_t \right) \right) \leq 0
\]
\[
\text{tr} \left( \mathbf{D}_t^\top \left( \mathbf{G}_j^{(t)} + \mathbf{D}_t \right) \right) \leq 0
\]
\[
\text{tr} \left( \mathbf{D}_t^\top \mathbf{G}_j^{(t)} \right) \leq - \left\| \mathbf{D}_t \right\|_F^2.
\]
which clearly shows that $D_i$ is a descent direction for all objectives.

### 6.8.3 Proof of Theorem 6

To prove the Theorems 6 and 7, we first need to show that by properly choosing the regularization parameter $\alpha$ our objectives are smooth. Recall that, our goal is to solve the following multi-objective optimization problem with non-convex components:

$$ f(U) = [f_1(U), \ldots, f_m(U)] $$

where $m = 1 + \binom{k}{2}$ with $k$ being the number of groups in the sensitive feature. Also, recall that in the case of fair PCA, we have $f_1(U) = -\frac{1}{2} \text{tr} \left( U^T X^T X U \right)$ is the overall reconstruction loss, and $f_i(U), i = 2, 3, \ldots, m$ are disparity errors for pair of groups. In what follows we use $\| \cdot \|$ and $\| \cdot \|_F$ to denote the spectral and Frobenius norms of a matrix, respectively.

To prove the theorem, we first show that all the the individual objective functions are smooth with bounded gradient (i.e., $f(\cdot)$ is component-wise smooth), conditioned that the regularization parameter $\alpha$ satisfies $\alpha \geq \max_{i,j} \gamma_d \left( X_i^T X_i \right) - \gamma_1 \left( X_j^T X_j \right)$ (recall that $\gamma_d(\cdot)$ is the smallest eigenvalue of input PSD matrix). To this end, we follow the definition of the smooth functions, i.e., $\| \nabla f(U) - \nabla f(V) \|_F \leq L \| U - V \|_F^2$.

In particular, for $f_1(U)$ we have:

$$ \| \nabla f_1(U) - \nabla f_1(V) \|_F = \| X^T X U - X^T X V \|_F $$

$$ \leq \| X^T X \| \| U - V \|_F $$

$$ \leq \gamma_{\max}(X^T X) \| U - V \|_F, $$

where the first inequality follows from the fact that for any two matrices $A$ and $B$ it holds that $\| A B \|_F \leq \| A \| \| B \|_F$, and the second inequality follows from the definition of spectral norm. The above inequality indicates that the objective corresponding to the overall reconstruction error is smooth with parameter $\gamma_{\max}(X^T X)$.

To show the smoothness of disparity errors, for simplicity, we only focus on one of the objectives between a single pair of sensitive features, say $s_i, s_j$, as the argument easily generalizes to other objectives/pairs due to symmetry. We also drop the subscript from function and use $f(U)$ to denote the regularized disparity error between groups $s_i$ and $s_j$.
defined as
\[
f(U) = \mathcal{E}_i(U) - \mathcal{E}_j(U) + \frac{\alpha}{2} \|U\|_F^2
\]
\[
= \mathcal{L}_i(U) - \mathcal{L}_i(U_i^*) - \mathcal{L}_j(U) + \mathcal{L}_j(U_j^*) + \frac{\alpha}{2} \|U\|_F^2
\]
\[
= -\frac{1}{2} \text{tr} \left( U^\top X_i^\top X_i U - U_j^\top X_j U \right) + \frac{1}{2} \text{tr} \left( U^\top X_j^\top X_j U \right) + \frac{\alpha}{2} \|U\|_F^2
\]
\[
+ \frac{1}{2} \text{tr} \left( U_i^* X_i^\top X_i U_i^* \right) - \frac{1}{2} \text{tr} \left( U_j^* X_j^\top X_j U_j^* \right)
\]
(6.40)

Following the definition of smoothness, we have
\[
\|\nabla f(U) - \nabla f(V)\|_F = \|X_i^\top X_i U - X_j^\top X_j U + \alpha U - X_i^\top X_i V + X_j^\top X_j V - \alpha V\|_F
\]
\[
= \left\| \left( X_i^\top X_i - X_j^\top X_j + \alpha I \right) (U - V) \right\|_F
\]
\[
\leq \|X_i^\top X_i - X_j^\top X_j + \alpha I\|_2 \|U - V\|_F
\]

Again, we can further upper bound the right hand side by using the definition of the spectral norm of a matrix:
\[
\|X_i^\top X_i - X_j^\top X_j + \alpha I\|_2 = \sup_{v \in \mathbb{S}^{d-1}} v^\top \left( X_i^\top X_i - X_j^\top X_j + \alpha I \right) v
\]
\[
\leq \gamma_{\text{max}} \left( X_i^\top X_i \right) - \gamma_{\text{min}} \left( X_j^\top X_j \right) + \alpha
\]
(6.41)

where \(\mathbb{S}^{d-1} = \{ x \in \mathbb{R}^d \mid \|x\|_2 = 1 \}\) is the sphere in \(d\) dimensions.

As a result, as long as the regularization parameter \(\alpha\) satisfies the following condition
\[
\alpha > \max \{0, \gamma_{\text{max}} \left( X_j^\top X_j \right) - \gamma_{\text{min}} \left( X_i^\top X_i \right) \}
\]
the disparity error objective between groups \(s_i\) and \(s_j\) is smooth with smoothness parameter \(\gamma_{\text{max}} \left( X_i^\top X_i \right) - \gamma_{\text{min}} \left( X_j^\top X_j \right) + \alpha > 0\). By symmetry, we can derive the smoothness condition for other pairs of groups as well, which results in the following condition on the regularization parameter:
\[
\alpha \geq \max_{i,j \in [k]} \gamma_{\text{min}} \left( X_i^\top X_i \right) - \gamma_{\text{max}} \left( X_j^\top X_j \right)
\]
to satisfy the smoothness of all objectives \(f_i(\cdot), i = 2, \ldots, m\). We note that one can use different regularization parameters for each pair depending on the eigen-gap between their covariance matrices as well.
We now turn to prove the convergence rate of the proposed algorithm to a Pareto fair subspace in general case as stated in (6.39), where we assume that the individual loss functions $f_i(U), i = 1, 2, \ldots, m$ satisfy Lipschitz continuous gradient condition (smoothness) with smoothness parameters $L_i, i = 1, 2, \ldots, m$. We also use $L$ to denote the maximum smoothness parameter, i.e., $L = \max_{i=1,2,\ldots,m} L_i$.

**Proof.** The proof begins by first bounding the difference in function values of each objective $f_i(U_t) - f_i(U^*)$, individually, following the convexity assumption:

$$f_i(U_t) - f_i(U^*) \leq \text{tr} \left( \left( \tilde{G}_i^{(t)} \right) ^\top (U_t - U^*) \right)$$

Then we can multiply both sides by $\hat{\lambda}_i$ and sum over $i$:

$$\sum_{i=1}^{m} \hat{\lambda}_i^{(t)} (f_i(U_t) - f_i(U^*)) \leq \sum_{i=1}^{m} \hat{\lambda}_i^{(t)} \text{tr} \left( \left( \tilde{G}_i^{(t)} \right) ^\top (U_t - U^*) \right)$$

$$= \text{tr} \left( \left( \sum_{i=1}^{m} \hat{\lambda}_i^{(t)} \tilde{G}_i^{(t)} \right) ^\top (U_t - U^*) \right)$$

$$= - \text{tr} \left( D_t ^\top (U_t - U^*) \right)$$

$$= \frac{1}{\eta} \text{tr} \left( (U_t - U_{t+1}) ^\top (U_t - U^*) \right)$$

$$= \frac{1}{2\eta} \left( \|U_t - U^*\|_F^2 + \|U_t - U_{t+1}\|_F^2 - \|U_{t+1} - U^*\|_F^2 \right)$$

$$= \frac{1}{2\eta} \left( \|U_t - U^*\|_F^2 - \|U_{t+1} - U^*\|_F^2 \right) + \frac{\eta L^2}{2}$$

where $\oplus$ follows from the smoothness assumption and definition of $L$. By summing up above inequality for all iterations $t = 1, 2, \ldots, T$ gives:

$$\sum_{t=1}^{T} \sum_{i=1}^{m} \hat{\lambda}_i^{(t)} (f_i(U_t) - f_i(U^*)) \leq \frac{1}{2\eta} \sum_{t=1}^{T} \left( \|U_t - U^*\|_F^2 - \|U_{t+1} - U^*\|_F^2 \right) + \frac{\eta L^2}{2}$$

$$= \frac{1}{2\eta} \left( \|U_1 - U^*\|_F^2 - \|U_T - U^*\|_F^2 \right) + \frac{\eta L^2 T}{2}$$

$$\leq \frac{1}{2\eta} \|U_1 - U^*\|_F^2 + \frac{\eta L^2 T}{2}$$

(6.42)

For the left hand side, since the $f_i(U_t)$ is a decreasing function by increasing $t$, we can
bound it by:

\[
\sum_{t=1}^{T} \sum_{i=1}^{m} \hat{\lambda}_i (f_i(U_t) - f_i(U^*)) \geq \sum_{i=1}^{m} \left( \sum_{t=1}^{T} \tilde{\lambda}_i^{(t)} \right) (f_i(U_T) - f_i(U^*)) \\
= \sum_{i=1}^{m} T \cdot \bar{\lambda}_i (f_i(U_T) - f_i(U^*)) ,
\]

(6.43)

where \( \bar{\lambda}_i = \frac{1}{T} \sum_{t=1}^{T} \tilde{\lambda}_i^{(t)} \). By plugging (6.43) back into (6.42) we have:

\[
\sum_{i=1}^{m} \lambda_i (f_i(U_T) - f_i(U^*)) \leq \frac{1}{2\eta T} R^2 + \frac{\eta L^2}{2} ,
\]

(6.44)

where \( \|U_1 - U^*\|_F^2 = R^2 \). By setting \( \eta = \frac{R}{L\sqrt{T}} \), the convergence inequality reduces to:

\[
\sum_{i=1}^{m} \bar{\lambda}_i (f_i(U_T) - f_i(U^*)) \leq \frac{RL}{2\sqrt{T}} ,
\]

(6.45)

We note that by setting \( \beta = \sqrt{T}/R \), the sufficient decrease condition in (6.15) is satisfied if the backtracking is employed. \( \square \)

### 6.8.4 Proof of Theorem 7

**Proof of Theorem 7.** The proof proceeds using the smoothness condition. In particular, for a smooth function \( f : \mathbb{R}^{d \times r} \mapsto \mathbb{R} \) with smoothness parameter \( L \) it holds that (descent lemma),

\[
f(V) \leq f(U) + \text{tr} (\nabla f(U), V - U) + \frac{L}{2} \|V - U\|_F^2 .
\]

From the backtracking line search we can find the learning rate at each step that gives us the maximum decrease. To that end, we will start from \( \frac{1}{2} \) and decrease it each time by half until all the objective have a maximum decrease defined in (6.15). Thus, if an \( \eta \) satisfies the condition the one step before that, \( 2\eta^* \), there is at least one objective not satisfying the condition. For instance, we consider the \( i \)th objective does not satisfy the condition with \( 2\eta^* \):

\[
f_i(U_t + (2\eta^*) D_t) \geq f_i(U_t) + \beta (2\eta^*) \text{tr} (D_t^T G_i^{(t)})
\]

(6.46)
Now from the Lipschitz continuity of the function as:

\[ f_i(U_{t+1}) \leq f_i(U_t) + \text{tr} \left( \nabla f_i(U_t) (U_{t+1} - U_t) \right) + \frac{L_i}{2} \| U_{t+1} - U_t \|_F^2 \]

\[ f_i(U_t + (2\eta^*) D_t) \leq f_i(U_t) + (2\eta^*) \text{tr} \left( \nabla f_i(U_t) D_t \right) + \frac{L_i (2\eta^*)^2}{2} \| D_t \|_F^2, \quad (6.47) \]

We proceed by combining (6.46) with this Lipschitz continuity (6.47) inequality which results in:

\[ \text{tr} \left( D_t^T G_i(t) \right) \geq -\frac{L_i \eta^*}{1 - \beta} \| D_t \|_F^2, \quad (6.48) \]

Also, from (6.38), we note that the left hand side term has a upper bound of \(-\| D_t \|_F^2\), implying

\[ \eta^* \geq \frac{1 - \beta}{L_i}, \quad (6.49) \]

which we can replace \( L_i \) with \( L_{\text{max}} = \max_{1 \leq i \leq m} L_i \), to obtain the lower bound on learning rate, that is \( \eta_t \geq C_1 = \min \{1, \frac{1 - \beta}{L_{\text{max}}}\} \). Hence, by the choice of learning rate, we know that at every step, we have the maximum decrease for every objective \( 1 \leq i \leq m \):

\[ f_i(U_{t+1}) \leq f_i(U_t) + \eta_t \text{tr} \left( D_t^T G_i(t) \right) \overset{\text{①}}{\leq} f_i(U_t) - \beta \eta_t \| D_t \|_F^2 \overset{\text{②}}{\leq} f_i(U_t) - \beta C_1 \| D_t \|_F^2 \]

where ① comes from Lemma 5 and (6.38), and ② follows from the bound on \( \eta_t \) in (6.49).

Summing up the last inequality for all iterations \( t = 1, \ldots, T \) and setting \( C = \beta C_1 \), we obtain:

\[ \sum_{t=1}^{T} C \| D_t \|_F^2 \leq \sum_{t=1}^{T} (f_i(U_t) - f_i(U_{t+1})), \quad (6.50) \]

In (6.50), the left hand side is greater than \( \min_{t=1,2,\ldots,T} CT \| D_t \|_F^2 \); and the right hand telescopes and can be upper bounded by \( M_u - M_l \), where \( M_u = \max_{i=1,2,\ldots,m} f_i(U_1) \) is the maximum value among objective functions at starting point and \( M_l \) is the lower bound on all objectives. Then the inequality becomes:

\[ \min_{t=1,2,\ldots,T} \| D_t \|_F \leq \sqrt{\frac{M_u - M_l}{CT}} \quad (6.51) \]

indicating that the proposed algorithm convergence to a Pareto stationary point (a point where the descent direction is 0 and none of the objectives can be further improved). \( \square \)
In this dissertation, two general multiobjective approaches are proposed to mitigate
the effects of bias in the training of machine learning models. These two approaches
can be adapted to different forms of bias. The first approach, targeted data-driven
regularization employs a small target dataset that is free of the bias, and it is utilized
in a bilevel programming to act as a regularization for the main learning procedure.
Using this approach, a weight for each group or class is assigned that is getting updated
during the training using that target dataset. The weights are updated in a way to
prevent the majority class or group to dominate the training. Empirical results for the
class-imbalance problem is presented, which shows the efficacy of this approach even in a
highly imbalanced case such as severe weather detection.

The second proposed approach, Pareto descent optimization is aiming at finding
Pareto-efficient points of the trade-off between the main learning objective and the bias
mitigation objectives. In this case, the task of bias mitigation is cast as a multiobjective
optimization problem, where using a gradient-based approach it is guaranteed to converge
to a Pareto stationary point of the problem. By extending this algorithmic solution to a
preference-based approach it converges to a point with the desired levels of compromises
between objectives. The proposed algorithmic solution is designed to trace the points
on the Pareto frontier before converging to that desired point. To showcase the efficacy
of this approach, it has been applied to the fairness problem in both supervised and
unsupervised settings. Empirical results of these two applications clearly indicate that
this approach outperforms other state-of-the-art algorithms in this field.

Delving into the details of the proposed algorithmic solutions, there are still some
questions left to be answered. Hence, we pose some of these questions as future directions to this research:

- Despite the practical success of the proposed targeted data-driven regularization, the convergence analysis of this algorithm is necessary to better understand the effects of different parts. For instance, the optimal size of the target dataset and the optimal number of inner and outer iterations for the stochastic approximation could be analyzed with careful convergence analysis.

- The setup introduced in [7, 3] for invariant risk minimization in the presence of spurious correlation is closely related to that of targeted data-driven regularization, when the target dataset is from different environments. An extension of targeted data-driven regularization to the invariant risk minimization problem would be an interesting future direction.

- The convergence analysis provided for the Pareto descent optimization is for the batch gradient descent. A useful extension to this work and in general to the multiobjective optimization domain is the analysis of the stochastic Pareto descent optimization, where the difficulty lies in showing at each step under what conditions the found direction would be a descent direction for all objectives.

- Federated learning [110, 62, 60, 61, 35, 36] is a novel distributed optimization approaches with decentralized data and computation that can be leveraged to learn a model from millions of devices without transferring their data to a central node. Since computations are done locally enforcing some of the methods proposed here might not be feasible in this setting. As it has been shown, this setting is highly prone to different forms of bias, and due to its special setting, where the computations are done locally, common bias mitigation approaches might not be practical. An interesting application of Pareto descent optimization is its extension to the federated learning settings for bias mitigation.

- Another form of biases is adversarial attacks. This form of bias has been vastly explored on a variety of studies, however, the trade-off between normal accuracy and robust accuracy is yet to be analyzed more carefully. In addition to this, the effects of adversarial nodes in a federated setting, where a coalition of nodes can intentionally manipulate the training process, and how it can be prevented are worthy of investigation. This is away from the generic adversarial learning, where the goal is to be robust against adversarial examples during the inference time. In
this setting, the goal is to prevent the damaging consequences of nodes producing adversarial examples during the training to degrade the global model’s quality.
In this appendix, the proposed framework for generating the Bow echo datasets used in Chapter 3 is discussed. This framework has been used to label bow echo samples on radar images for two years of 2008 and 2009.

A.1 Introduction

Monitoring and storing climatic data around the globe provide a vast amount of data for weather condition analysis. In spite of the fact that computational power is emerging continuously, automatic severe weather forecasting is costly and not always accurate. Meteorologists leverage various and complex models to forecast storms using data from a collection of sensors, including tools and data at the Storm Prediction Center (SPC)\(^1\) of the National Oceanic and Atmospheric Administration (NOAA). The data gathered from these sensors are stored historically; hence it can be leveraged to extract historical patterns of different severe weather conditions. Although meteorologists have developed numerous and complicated models for forecasting storms, they still rely significantly on their interpretations instead of automated algorithms. Further, the majority of these models depend on initial conditions and are highly sensitive to noise, making forecasting difficult. Therefore, it is inevitable for this field to combine big data, computer vision, and data mining algorithms with these models to seek faster, more robust, and more accurate results.

Severe weather conditions consist of thunderstorms, tornadoes, floods, lightning, hail, 

\(^1\)http://www.spc.noaa.gov
and strong winds. Each of these conditions are investigated widely in meteorological literature, and they need different sources for detection and forecasting, such as satellite images, radar images, temperature, air pressure and wind speed, to name but a few. These events are the primary causes of a large amount of damage around the globe. For instance, according to the National Severe Storms Laboratory (NSSL)\(^2\), damaging winds or *straight-line winds* are the major causes of nearly half of all reports of severe weather conditions in the United States. These winds can reach the speed of 100 miles per hour and have a damage path up to hundreds of miles. Bow echoes, convective line segments with an archery bow shape, are mainly associated with these strong straight-line winds. In some cases, parts of bow echoes can form tornadoes and new thunderstorms. Hence, bow echo detection can be used as a way of forecasting such destructive severe weather conditions. Accurate and on-time forecasting of these events seems necessary and would help to mitigate damages.

As Klimowski et al. [96] found 273 cases of bow echoes between 1996 and 2002, it seems to be a common pattern among weather patterns. Their investigations revealed that bow echoes are causing nearly 33% of severe convectively generated winds in the U.S. [95]. NSSL in partnership with other organizations performed a field experiment on bow echoes called *Bow Echo and Mesoscale Convective Vortex Experiment (BAMEX)* to investigate bow echoes and extremely damaging surface winds with them in more detail [33]. Although, meteorologists have done quite number of research studies on bow echoes and their effects [95, 96, 33, 127], there is no evidence of computer-aided algorithms in bow echo detection and forecasting in the literature. While there is no machine learning and computer vision frameworks for detecting bow echoes, there has been some studies investigating severe weather conditions using satellite images or other tools. Zhou et al. [159] build a framework to estimate cloud motions on satellite images especially on hurricanes when clouds form a cyclic or comma shape. Zhang et al. [158, 157] use optical flow algorithms [68] on satellite images to predict the location of thunderstorms. However, radar images are different from satellite images in nature representing different information. Hence, the approach to extract a pattern in one of them is not necessarily applicable to the other. Narasimhan and Nayar [116] model severe weather conditions and their effect on the quality of urban images and videos and how to restore them using the structure and depth discontinuities in the scene. The main difference of this research from ours is that the radar images do not have the notion of depth in their images and the structure of the whole image would change during

\(^2\)http://www.nssl.noaa.gov
the time. Quinan and Meyer [128] build a system to visualize ensemble of forecasts for meteorologists using an open source platform. Additionally, the shorter abstract of this paper is printed in [78].

In this appendix we propose a method for detecting bow echoes in radar images. This bow shape signature of bow echo leads us to use computer vision algorithms for particular shape detection and matching. In this regard, we first use radar images and develop our algorithm for skeletonization, which then can be used in the shape matching algorithm through our suggested descriptor, *Skeleton Context*. Finally, we use *Mixture Discriminant Analysis (MDA)* to classify bow echo shapes in radar images.

### A.2 Bow Echo Feature Extraction

Severe weather conditions such as thunderstorms, tornadoes, hail, and especially strong straight-line winds are associated with bow echoes. The wind with a bow echo can be fierce and reach violent intensity. The term bow echo was coined by Fujita [54], to describe strong outflow winds associated with storms that spread out in straight lines over the ground. Przybylinski categorizes bow echoes in two categories [127]: (i) bow echo patterns associated with derechos or straight-line winds and (ii) bow echo patterns associated with vortices, including tornadoes. Klimowski et al. [96] classify different types of bow echoes and their evolution from meteorologists’ point of view. They start with a radar echo and then evolve into a bow echo. In this proposed framework, we aim to use this topological feature of these phenomena to detect them using computational approaches.

Our proposed scheme for detecting bow echoes, shown in Figure A.1, consists of two main steps, skeleton extraction and matching. In the first step, we take a radar image and extract its regions of interest (parts that we can find bow echoes). Then using our skeletonization and skeleton pruning framework, to extract their skeleton. In the second step, using our suggested shape descriptor, skeleton context, to extract features for shape matching part. After matching those parts to a bow echo prototype, based on the distance between them and their matched bow echo prototype, we are able to identify whether they are a bow echo or not.
Figure A.1: The complete scheme for automatic bow echo detection. In the first stage, it extracts regions of interest, which are red parts in radar images. Then in the skeletonization step, it extracts skeleton map of each part, and prunes it using the proposed algorithm. After finding pruned skeleton, it computes skeleton context and finds the nearest match in the database of bow echo prototypes. Finally, it uses mixture discriminant analysis classifier to detect whether it is a bow echo or not.

A.2.1 Radar Images and Regions of Interest

Our dataset consists of images from NEXRAD level III radar of National Weather Service (technical name WSR-88D), which can measure precipitation and wind movement in the atmosphere. These images are gathered from 160 active high resolution radar sites around the U.S. continent. We use base reflectivity images from NEXRAD level III radar, which represent the amount of returned power to the radar from transmitted signal after hitting precipitation in the atmosphere. The images have 4-bit color map with 6,000 × 2,600 pixels of spatial resolution, which are stored every five minutes [72]. That is, in a whole year there are more than 105,000 images with mentioned quality. The color map associated to these radar images is shown in Figure A.2 having the range from 0 dBZ to 75 dBZ for reflectivity. The range of the reflectivity from 0 dBZ to −30 dBZ, alongside with “No Data” regions (due to spots with beam blockage in the mountains and outside of the U.S.) is represented by a black color.

Bow echoes can be spotted in heavy precipitation red regions on radar images (i.e., with reflectivity of higher than 50 dBZ). As shown in Figure A.2, the bow echo happened on May 2 in 2008 over Kansas City. Hence, in searching for bow echoes in radar images,
regions of interest are red in color ($\geq 50$ dBZ). To extract these parts we can set a threshold on their RGB values, but it would result in patchy areas and not connected regions. Although human eyes can cluster them as a unified region, computer algorithms need to perform a pre-processing in order to connect those parts together. We use some morphological operations such as image closing along with active contours to improve the extraction of the connected components.

### A.2.2 Skeletonization

Skeleton of a shape is a low-level representation that can be used for matching and recognition purposes in various fields of study including image retrieval and shape matching [143] or human pose estimation and recovery [67, 46, 47]. Skeleton can provide a good abstraction of a shape, which contains topological structure and its features. Because it is the simplest representation of a shape, there has been an extensive effort among researchers to develop generic algorithms for skeletonization of shapes [119, 140, 89, 37, 130]. However, Saha et al. [133] claim that since there is no “true” skeleton defined for an object, the literature in skeletonization lack of a robust evaluation. The vast majority of the algorithms are based on Blum’s “Grassfire” analogy and formulation for skeletonization [19]. The most important key factor in skeletonization algorithms is to preserve the topology of the shape. Bai et al. [9] introduce a convexity constraint for skeletonization and pruning that is used in contour partitioning with discrete curve evolution, and then use this skeletonization to match the skeleton graphs [8]. However, since their method for skeletonization and pruning removes unwanted branches iteratively,
it might miss some parts of the main skeleton, and hence, cannot preserve the topology of a shape completely. One of the most widely used algorithms is based on measuring the net outward flux by using Euclidean Distance Transform (EDT) of the binary image followed by a topology preserving thinning algorithm [37]. We use the method introduced by Dimitrov et al. [37] to calculate the net outward flux per unit area and detect the location of the pixels where conservation of energy principle is violated. EDT maps a binary image into a gray level image with value of each pixel represents its euclidean distance to the border of image. Given Euclidean distance of an image ($D_E$), we should first compute the gradient vector field ($\nabla D_E$), and then the divergence of this vector field [37]. Mathematically, the divergence of the gradient vector field ($\nabla \cdot (\nabla D_E)$) is defined as the limit of the net outward flow of the field across the boundary of the area around the given point, while the area is shrinking to zero:

$$\nabla \cdot (\nabla D_E) = \lim_{S \to 0} \iint_C \frac{\nabla D_E \cdot n}{S} dC ,$$  \hspace{1cm} (A.1) 

where $C$ is the boundary, $S$ is the area, and $n$ is the normal vector of the boundary. Hence, we can calculate net outward flux at each point $P = (x, y)$ as follow:

$$\text{Flux}(P) = \sum_{i=1}^{8} \nabla D_E(Q_i) \cdot n ,$$  \hspace{1cm} (A.2) 

where $Q_i$’s are neighbor points to point $P$. According to the direction of the normal vector, we can determine that positive or negative flux values are representing drain or source of energy, where energy-draining points are internal skeletal points, and energy-generating points are external skeletal points. In the Figure A.3(b), the EDT of the binary image, in Figure A.3(a), is shown. Then, in Figure A.3(c) the net outward flux for this Euclidean distance transform is computed.

By setting a threshold on flux values, initial binary skeleton map can be computed as depicted in Figure A.3(d). Because we are dealing with highly boundary-variant shapes, the skeleton map would contain a large number of unwanted branches that make the subsequent matching steps complicated. We need to develop an automated method to remove all such branches while keeping the main skeleton intact. We introduce a new method to prune the skeleton using fuzzy logic, which will be discussed in the next section. Our pruning algorithm needs to have the complete graph information of the skeleton including its vertices and edges’ pixels coordinates. Therefore, the skeleton map is converted to a graph before the pruning step.

129
Figure A.3: Different stages in the skeletonization process. (a) We start from extracting the binary image of the object. (b) We apply Euclidean Distance Transform on the binary image. (c) Using (A.2) calculate the flux map. (d) Apply a threshold to extract the initial skeleton. (e) and (f) are the output of Algorithm 5, which extracts vertices and edges of the initial skeleton.

Converting a skeleton map to a graph consists of two steps, namely extracting vertices and then points of each edge. We will go through each step as follow:

**Extracting Vertices:** For finding the vertices of the skeleton graph we need to scan the whole pixels of the image and based on the structure of other pixels around each pixel, we could decide whether it is a vertex or not. In this regard, we extract a $3 \times 3$ matrix around each pixel, which consists of its 8-connected neighbors. Then, in this matrix we build a local graph having pixels with value of “1” as its vertices. The connectivity of this graph is 4-connected neighbors, which means that 2 vertices are connected if and only if they are in one of the 4 main directions of each other (left, right, up, down). After that, in this graph we could calculate the Euler characteristic using its number of vertices and edges, as it is computed in Algorithm 5. If the Euler characteristic of the graph is greater than 2, it shows that this point is on the crossing of three or more
edges, hence, it is a vertex with a branch type. If it is 1, it means that this point is an end point vertex of an edge. When it equals 2, the point is a simple edge point and not a vertex. Vertices extracted in this way for the skeleton in Figure A.3(d), is depicted (with a magnified part for a better representation) in Figure A.3(e). Red points in the image are branch points, and green ones are end points.

**Extracting Edge Points:** After finding vertices in the graph, we should form the edge list of the graph. We start with a random end point and traverse its neighbors to reach to a branch point or other endpoints. If the neighbor pixel is not a branch point nor an endpoint, it would be added to the current edge’s pixel list. When we reach a branch point, we should add another edge to the edge list, having that branch point as its first point, and start searching edge points for the next edge in the edge list. If we reach an end point, we just start searching edge points for the next edge in the edge list. This approach will be continued until there is no edge left unprocessed. Detailed algorithm of transforming skeleton map to graph is in Algorithm 5, and the result of finding edge list is shown in Figure A.3(f) with different colors for different edges.

### A.2.3 Skeleton Pruning using Fuzzy Logic

Having high sensitivity to border variations, almost all algorithms for skeletonization need to be followed by a pruning stage in order to remove thin branches caused by boundary deformations. These branches may significantly change the skeleton graph, and hence they should be treated carefully for the matter of topology preserving in skeletonization algorithms. This issue would be intensified in case of radar images and bow echo shapes, because they have drastic variations on their borders, as it is evident in radar images. There has been studies [9, 103, 138] investigating direct as well as indirect methods to address this issue. Most of these algorithms use Boolean logic in their decision to remove or keep the branches. The output of these algorithms is a crisp value attributed to each edge distinguishing branch edges from main skeleton edges. However, if we ask a person to do pruning on a skeleton graph, he or she would extract the main skeleton with an uncertainty to some extent. On the other hand, fuzzy logic introduces many-valued logic in close proximity to human decision making system [152, 76]. Hence, we propose an approach based on fuzzy inference system to prune skeleton graph and extract the main skeleton.

In our method, we use the outward flux values of the pixels as an input to the fuzzy inference system. Heuristically from raw images of flux values (Figure A.3(c)), the higher the value of the outward flux in each pixel, the more probable that the pixel is in the
Algorithm 5: Skeleton2Graph

input Skeleton Map

function FindVertices(Skeleton):
    for Point on Skeleton do
        PointMatrix ← find 8-Connected neighbors
        Filter PointMatrix by a 4-connected neighbors mask
        EulerCharacteristic ← #Vertices − #Edges
        if EulerCharacteristic > 2 then
            BranchPoints ← Point
        else if EulerCharacteristic = 1 then
            EndPoints ← Point
        end
    end
    return BranchPoints, EndPoints

function GenerateEdgeList(Skeleton, BranchPoints, EndPoints):
    EdgeList{1} ← Select one End Point randomly
    EdgeNumber ← 1 while EdgeNumber ≤ #Edges in EdgeList do
        SearchPoint ← EdgeList{EdgeNumber}(end)
        SearchMatrix ← 8-Neighbors Connected to SearchPoint
        for Point in SearchMatrix do
            Set value to −EdgeNumber
            if Point is in BranchPoints then
                Add a new edge to EdgeList with having this Branch point as its first Point
                EdgeNumber ← EdgeNumber + 1
            else if Point in EndPoints then
                Point ← EdgeNumber + 1
            else
                Add Point to EdgeList{EdgeNumber}
            end
        end
    end
    return EdgeList

main skeleton. Therefore, based on this observation we can extract a feature for every edge connected to a vertex in the skeleton graph. In order to record vertices properties in the flux images, we form an array, called $\Gamma$, for each vertex with the length of the number of the edges linked to it. The value attributed to each edge $e_j$ connected to the vertex $V_i$ could be computed as follow:

$$\Gamma\{V_i, e_j\} = \frac{1}{M_j - 1} \sum_{P_j=2}^{M_j} (W_G(P_j) \cdot \text{Flux}(P_j)),$$  \hspace{1cm} (A.3)
in which, \( j = 1, ..., N \), indicating the \( j \)th edge connected to \( V_i \). \( N_i \) is the number of edges linked to \( V_i \), \( M_j \) is the number of pixels in the \( j \)th edge, \( P_j \) is the index of pixels on the \( j \)th edge, starting from the vertex \( V_i \), and \( W_G(P_j) \) is the Gaussian weight for each pixel of edge \( e_j \) computed as follow:

\[
W_G(P_j) = \exp \left( -\frac{||P_j - V_i||^2}{2\sigma^2} \right).
\] (A.4)

Our proposed fuzzy inference system (FIS) consists of two components:

- **FIS-1**: Fuzzy inference system to compute degree of belief of each pixel to main skeleton edges.
- **FIS-2**: Fuzzy inference system to compute degree of belief of each pixel to branch edges.

The FIS-1 output, indicates that to what extent we believe an edge belongs to main skeleton. Afterwards, we use this value as an input to FIS-2 to compute the extent to which that we believe an edge belongs to branch edges. These values are the same for the pixels of the edge and varies among different edges. In following subsections we introduce these two fuzzy inference systems, their inputs, rules, and outputs. And then we go through the details of our algorithm for pruning skeleton.

### A.2.3.1 Main Skeleton Fuzzy Inference System

We now describe the inputs, the outputs, and the operators used in the first fuzzy inference system (FIS-1). In this inference system, unlike the second one, because keeping the main skeleton correctly in the pruning step is more important than omitting branches, we need to have more resolution in the definition of the inputs and outputs. Accordingly, we use more number of membership functions with Gaussian type, that decay faster than linear, in this inference system, compared to the fewer number of membership functions with trapezoid type in the second one.

**Fuzzy Inputs**: FIS-1 has two inputs as follows:

- **Importance Value \((I)\)**: which indicates the importance of each vertex, and is generated from vertices properties of linked edges. The details of computing importance value would be described later. Based on variations in the importance values of different vertices, we can calculate the expected value and variance of this feature in an image. After statistically analyzing the expected value \((\mathbb{E}\{I\})\) and variance \((\sigma I)\) of the importance
value of all vertices, we come up with three Gaussian membership functions with their center on \([\mathbb{E}\{I\} - 6\sigma_I, \mathbb{E}\{I\}, \mathbb{E}\{I\} + 6\sigma_I]\) and standard deviation (sigma) equal to \(\sigma_I\). We call these membership functions Low, Medium, and High, according to their centers.

**Edge Length \((L_E)\):** One of the most important features for detecting main skeleton edges, is edge length. However, branches mostly happen where boundary is deformed with variations, and hence the skeleton would be furcated into too many small branches. As a result, we can use the edge length as an indicator for main skeleton pixels, alongside with other factors. Edge length in each image is random variable that we can find its expected value as \(\mathbb{E}\{L_E\}\), and its standard deviation as \(\sigma_{L_E}\). Again with statistical analysis of edges, we attribute three different Gaussian membership functions with their centers on \([\mathbb{E}\{L_E\} - 5\sigma_{L_E}, \mathbb{E}\{L_E\}, \mathbb{E}\{L_E\} + 5\sigma_{L_E}]\) and standard deviation (sigma) equal to \(\sigma_{L_E}\). These membership functions are called Small, Medium, and Long respectively. In this case, we should make sure that the minimum value for edge length is not less than zero.

**Fuzzy Output:** This FIS has one output, that is, *Main Skeleton Degree of Belief* \((\Psi_{MS})\). This output represents the degree of belief on pixels to be on the main skeleton graph. The range of its value is between \([0, 1]\), and we define 5 different Gaussian membership functions as its fuzzy sets. The sigma value for these fuzzy membership functions is set to 0.05 and their centers are on \([0, 0.25, 0.5, 0.75, 1]\). These functions are named as follow: Very Low, Low, Average, High, and Very High. The output resulted from this FIS for the skeleton in Figure A.3(d) is shown in Figure A.8(a). The higher the value in image, the higher degree of belief of main skeleton \((\Psi_{MS})\) on that edge. Fuzzy membership functions of the inputs and the output of this inference system for a sample skeleton is depicted in Figure A.4.

**Fuzzy Operators:** In each fuzzy inference system we should define methods of integration of membership functions. First of all we should decide about the method of integrating different inputs in each rule, then the method for implication of the output in each rule, and at the end the method of aggregation of outputs from each rule. Hence, we choose these operators as follow:

- **Fuzzy Operation:** we choose the simplest method, that is, \(\text{min}\) for AND operations and \(\text{max}\) for OR operations.

- **Implication Method:** for the implication of the output we choose \(\text{min}\) operator.

- **Aggregation:** for the aggregation of the outputs, we use \(\text{max}\) operator.
A.2.3.2 Branch Fuzzy Inference System

The details of the second fuzzy inference system (FIS-2) are provided below.

**Fuzzy Inputs**: FIS-2 has three inputs including the output of FIS-1.

- **Main Skeleton Degree of Belief ($\Psi_{MS}$)**: This is the output from the first FIS. However, for this FIS, as it was discussed before, we only consider two fuzzy membership functions, namely, *Low* and *High*. Instead of Gaussian, we choose trapezoid membership functions, because it would result in more fuzziness than before inherited from the definition of branches as well. The parameters of these two membership functions are $[0, 0, 0.3, 0.55]$ and $[0.4, 0.7, 1, 1]$, where the four parameters $[a, b, c, d]$ define a trapezoid membership function uniquely as follow:

$$mf(x, [a, b, c, d]) = \max \left( \min \left( \frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right).$$  (A.5)

- **Edge Length ($L_E$)**: Because of the importance of edge length in distinguishing between main skeleton and branch edges, we use this feature in our second FIS, with the same range for its universe of discourse. But we merely define two membership functions in FIS-2 for this input, consisting of two trapezoid functions named *Small* and *Long*, in which their parameters are $[E\{L_E\} - 5\sigma_{L_E}, E\{L_E\} - 5\sigma_{L_E}, E\{L_E\} - 4\sigma_{L_E}, E\{L_E\} + \sigma_{L_E}]$ and
\[
[\mathbb{E}\{L_E\} - \sigma_{L_E}, \mathbb{E}\{L_E\} + 2\sigma_{L_E}, \mathbb{E}\{L_E\} + 5\sigma_{L_E}], \quad \text{respectively. They are not symmetric because the distribution of edge length is more accumulated near zero.}
\]

Curvature Score \((S_C)\): In the course of searching for the main skeleton, we may encounter with a vertex that has two output edges, in which almost all of their properties are similar. The only difference between these two edges is their angle with the nearest main skeleton edge. The more this angle is close to zero, the more probable that we categorize the edge as a main skeleton edge rather than a branch edge. Therefore, we introduce Curvature Score for vertex \(V_i\) as follows:

\[
S_{C}^{V_i}(e_{i_1}, e_{i_2}) = \cos(\theta_{i_1,i_2}) = \frac{\langle e_{i_1}, e_{i_2} \rangle}{||e_{i_1}|| \cdot ||e_{i_2}||}, \quad (A.6)
\]

where \(e_{i_1}\) is the vector starting from the middle point of reference edge (or main skeleton edge) to \(V_i\), and \(e_{i_2}\) is the vector starting from \(V_i\) to the middle point of the test edge. The universe of discourse for this input would lie in the range of \([-1, 1]\), and we choose two trapezoid membership functions on this range with the name of Averted and Straight. The parameters of these two membership functions are \([-1, -1, -0.7, 0.2]\) and \([-0.1, 0.7, 1, 1]\), respectively.

Fuzzy Output: The fuzzy output of this FIS is representing the degree of belief on edges to be member of branch edges, which is in the range of \([0, 1]\), and we call it Branch Degree of Belief \((\Psi_B)\). We consider three membership functions for this output, namely, Low, Average, High, in which the first and the last one are trapezoid and the second one is a triangle membership functions with parameters \([0, 0, 0.2, 0.4]\), \([0.4, 0.5, 0.6]\), and \([0.6, 0.8, 1, 1]\), respectively. The result of this FIS on the skeleton of Figure A.3(d) is depicted in Figure A.8(b). Higher values in the image shows higher branch degree of belief \((\Psi_B)\). Fuzzy membership functions of the inputs and the output of this inference system for a sample skeleton is depicted in Figure A.5.

Fuzzy Operators: We use the same fuzzy operators as the first FIS for this FIS, that is, \(\text{min}\) for AND operation and \(\text{max}\) for OR operations in the rules, \(\text{min}\) operator for output implications, and \(\text{max}\) operator for aggregations.

A.2.3.3 Importance Value

As it was mentioned in FIS-1, we have to compute value \(I\) for each vertex based on their flux values. Basically, this measure computes the importance level of each vertex or path based on its subsequent edges’ flux values, plus its own flux value. To calculate
the importance value of the vertices, we consider the skeleton graph as a directed tree starting from the best edge, with respect to the flux values of its pixels, as its root and all other edges connected to its vertices in their spatial order. Finding the best edge which has the highest flux value (the average of $\Gamma$ values of its both vertices), we continue to move from both ends of the best edge toward other edges, until all the edges are covered. Because we are computing the importance value of each vertex in a predefined direction, it could be considered as two “Directed Tree”s (from both ends of the best edge) with vertices and edges in a hierarchical manner. Hence, in the course of computing importance value of a vertex we could include flux values of edges in the lower level linked to that vertex in the hierarchy. Sometimes flux values of edges on the main skeleton decrease (in proportion to their linked edges), which leads to a false detection by pruning algorithm, if we merely rely on flux values of each edge. Thus, this operation ensures to choose the main skeleton edges by increasing their importance values. As a result, we
want to add depleted version of importance value of lower level vertices to importance value of current vertex. Hence, we can calculate the importance value for the vertex \( V_i \) in the recursive scheme as follows:

\[
I(V_i, \Omega) = \begin{cases} 
C_d \times \sum_{k_i=1}^{M_i,\Omega} I(V_k, \Omega - 1), & \Omega > 0 \\
C_d \times \sum_{k_i=1}^{M_i,\Omega} \sum_{j_i=1}^{N_i,\Omega} \Gamma \{V_k, e_{j_i}\}, & \Omega = 0 \text{ or } V_k \in \text{EndPoints} 
\end{cases} 
\tag{A.7}
\]

in which, \( \Omega \) shows the depth level needed to be included in the calculation of importance value for that vertex. \( C_d \) is a damping factor between 0 and 1, \( M_i,\Omega \) is the number of vertices linked to \( V_i \) in the level \( \Omega \), and \( N_i,\Omega \) is the number of edges linked to \( V_i \) in the level \( \Omega \) (note that this connection might be indirect). In summary, for calculating the importance value of a vertex we should consider vertices properties (\( \Gamma \)) of all subsequent vertices below this vertex in the tree. For further clarification, consider a simple skeleton graph in Figure A.6, with 13 vertices and 11 edges. Assume that we found the best edge in the sense of flux value, and we want to calculate the importance value of vertex \( V_1 \) with \( \Omega = 2 \), hence, we should consider 2 layers of vertices lower than \( V_1 \) in the hierarchical graph depicted in Figure A.6. These vertices are colored green on the figure, therefore based on (A.7), we could calculate this value:

\[
I(V_1, \Omega = 2) = C_d \times (I(V_3, \Omega = 1) + I(V_4, \Omega = 1)), 
\tag{A.8a}
\]
\[
I(V_3, \Omega = 1) = C_d \times \Gamma(V_3, e_{1-3}), 
\tag{A.8b}
\]
\[
I(V_4, \Omega = 1) = C_d \times (I(V_6, \Omega = 0) + I(V_7, \Omega = 0)), 
\tag{A.8c}
\]
\[
I(V_6, \Omega = 0) = C_d \times \Gamma(V_6, e_{4-6}), 
\tag{A.8d}
\]
\[
I(V_7, \Omega = 0) = C_d \times \Gamma(V_7, e_{4-7}), 
\tag{A.8e}
\]

where you can see the recursive property of importance value. Thus, to find the importance value of vertex \( V_1 \), we should calculate the importance values of \( V_6 \) and \( V_7 \) from (A.8d) and (A.8e), and substitute them in (A.8c). After calculating the importance value of \( V_3 \) and \( V_4 \) from (A.8b) and (A.8c), we can substitute them in (A.8a) and calculate the importance value of \( V_1 \). Note that as we go deeper in lower levels, the effect of their importance values is decreased because of \( C_d \).
Figure A.6: A simple skeleton graph with 13 vertices and 11 edges. Colored vertices are included in the calculation of the importance value of $V_1$ when $\Omega = 2$.

A.2.3.4 Pruning Algorithm

Using fuzzy inference systems with proper importance values of vertices and edges, the functionality of the pruning algorithm would be straightforward. We just need to define some parameters to give users ability to decide to what extent they tolerate branches in an image and the maximum threshold of $\Psi_B$ on an edge, to consider it as a branch. We define these parameters as follows:

$$
B_T \equiv \text{Branch Tolerance} \in [0, 1], \\
\Psi^*_B \equiv \text{\Psi_B Threshold} \in [0, 1], 
$$

(A.9)

where $B_T = 0$ means we do not tolerate any branches and we just want the medial axis. For example, if $B_T = 0$ and $\Psi^*_B = 0.6$, then we would like to get rid of all edges that have $\Psi_B$ greater than or equal with 0.6. Since by setting the $\Psi_B$ threshold to 0.6, it means that we consider all edges on skeleton with $\Psi_B$ greater than 0.6 as a branch. If $B_T$ is set to 1, it does not mean that we want to keep all edges with $\Psi_B$ less than $\Psi^*_B$, but we want to constraint the decision based on the situation of each branch point. Take for instance, a branch point has two edges with both $\Psi_B$ values less than $\Psi^*_B$ and $B_T = 1$, but there is a large difference between their values, which makes us to choose the one with the lesser value. Hence, based on these two parameters and $\Psi_B$ values in each branch point separately, we should form a Belief Window ($BW$) to filter desired values.
on that particular branch point. This window should always start from the minimum value among $\Psi_B$ values of edges linked to a branch point, and can have a size in the form of equation below for the vertex $V_i$ and set of edges linked to it in the lower level $E_i$:

$$BW(V_i, E_i) = B_T^{\Delta(V_i, E_i)} \times \Delta(V_i, E_i)^{f_1(B_T)} \times e^{f_2(\Delta(V_i, E_i))},$$

(A.10)

where

$$\Delta(V_i, E_i) \triangleq \Psi_{B}^* - \min_{E_i} \Psi_B,$$
$$f_1(B_T) \triangleq \alpha + \beta B_T,$$
$$f_2(\Delta(V_i, E_i)) \triangleq \kappa + \lambda \Delta(V_i, E_i).$$

(A.11)

Hence, in this schema we always keep an edge with the lowest branch degree of belief ($\Psi_B$) in each branch point. Based on parameters defined by the user we form a belief window, which can decide whether we should keep other edges or not. This function would satisfy aforementioned property of belief window for choosing branches. Parameters of lines in $f_1$ and $f_2$ can be set heuristically. For example, we can set them as follow: $\alpha = 1.5$, $\beta = 0.2$, $\kappa = 1.5$, and $\lambda = -2.5$. Changes in the size of belief window with changing $\Delta$ and $B_T$ is depicted in Figure A.7. This window size generally would increase when the $\Delta$ increases. The slope of this change is high when $\Delta$ is small, however, as the gap between $\Psi_{B}^*$ and minimum branch degree of belief in a branch point goes high, it decreases. When this window is formed, in each branch point we can decide which edges to keep or omit. Detailed pruning algorithm using fuzzy logic is in Algorithm 6, which takes a skeleton graph and its associated flux map, and calculate the pruned skeleton graph. It starts from one edge and try to find edges that should be kept based on the criteria discussed before. The pruned skeleton of Figure A.3(d) with zero branch tolerance is in the Figure A.8(c).

### A.2.4 Skeleton Context

**Shape Context** [15] as a powerful shape descriptor represents a rough distribution of all other points with respect to a selected point in terms of distance and angle. It has quite number of applications in object recognition [132], pose recognition [139], animation construction [151], to name but a few. Shape context is used to find correspondences between samples from border of two shapes, and then find the cost of matching two shapes using bipartite graph matching. After that, parameters for an affine transform
Figure A.7: Belief Window (BW) versus \( \Delta \), and \( B_T \). The slope of change in BW size would decrease by increasing \( \Delta \). This would be intensified by decreasing branch tolerance (\( B_T \)).

Figure A.8: Output of pruning algorithm: (a) main skeleton degree of belief, (b) branch degree of belief, (c) Pruned skeleton. In (a) and (b) images are in grayscale, and higher values represent higher degree of beliefs are extracted using thin plate spline (\( TPS \)), in order to map points from one shape to their correspondences in the other shape with warping the coordinates. Finally, a notion of shape distance for recognition purposes is exploited.

As there are a lot of fluctuations over the boundaries of the shapes in radar images, these make object matching with boundary samples less effective, and it may result in false matching. On the other hand, matching objects using skeleton samples sounds more robust in the sense that pruned skeleton contains complete shape topology regardless
Algorithm 6: Pruning

function Pruning(SkeletonGraph, Flux):
    RootEdge ← Find the best edge with highest VP values
    PointsQue ← Vertices(RootEdge)
    QueNumber ← 1
    while QueNumber ≤ # Points in PointsQue do
        $V_i$ ← PointsQue(QueNumber)
        $E_i$ ← Edges linked to $V_i$ in a lower level
        forall Edges in $E_i$ do
            $\Psi_{MS_j} ← \text{FIS-1}(V_i, E_i_j)$
            $\Psi_{B_j} ← \text{FIS-2}(V_i, E_i, \Psi_{MS_j})$
        end
        BW ← Form the BW based on (A.10)
        Choose candidate edges based on BW and their $\Psi_{B_j}$ values
        Add end points of the candidate edges to PointsQue
        QueNumber ← QueNumber + 1
    end
    return PrunedSkeleton

of its boundary variations. Hence, we use shape context to introduce a new descriptor called skeleton context. As it is shown in an example in Figure A.9, skeleton context is log-polar histogram, formed for each sample point on the skeleton. For each sample point $P_i$, the center of this log-polar histogram is located on that sample point, then each bin in the histogram represents the number of sample points in the specific angle and range of distance from the center (i.e. $P_i$) determined by that bin. We use the notation of $H_{SC}(P_i, r_{k1}, \theta_{k2})$, to show the value of skeleton context’s histogram for point $P_i$, in the $(r_{k1}, \theta_{k2})$ bin. For instance, when $H_{SC}(P_i, r_m, \theta_n) = 10$, it means that in the distance range of $r_{m-1} \leq r < r_m$ and in the angle range of $\theta_{n-1} \leq \theta < \theta_n$ from the point $P_i$ in the skeleton, there are 10 other sample points. Basically, these histograms for each point visualize the distribution of other points on the skeleton with respect to that point, and hence it could play the object descriptor role for the shape matching purposes. As a result, we could use these descriptors as the feature data for bow echo detection in the next step. The skeleton context can be calculated using the following equation:

$$H_{SC}(P_i, r_{k1}, \theta_{k2}) = |\text{Bin}(P_i, r_{k1}, \theta_{k2})|,$$

$$\text{Bin}(P_i, r_{k1}, \theta_{k2}) = \left\{ X \in \mathbb{S} \mid r_{k1-1} \leq \|X - P_i\|_2 < r_{k1}, \bigcap \theta_{k2-1} \leq \angle(X, P_i) < \theta_{k2} \right\},$$

(A.12)
where $|.|$ shows the number of members in a set, $P_i$ is a sample point on the skeleton that we want to calculate its skeleton context, $S$ is the set of sample points in the skeleton, and $\angle(X, P_i)$ calculate the angle of a vector from $P_i$ to $X$ when $P_i$ is on the origin of coordinates, with respect to the horizontal coordinate. In Figure A.9 the skeleton context is computed for two points of two objects, which matched together in our algorithm.

**A.3 Bow Echo Classification and Detection**

With skeleton context defined in previous section, we are able to extract features from objects in radar images and use them in the recognition process. In the following subsections, we introduce our proposed model for learning features of bow echoes and implementing a classifier in order to detect bow echoes in tons of objects extracted from radar images. First, we introduce four indicators that can help us to define distance metric between two skeletons. Then, we impose neighboring cost to enforce partial shape matching. In the end, we use the defined distance metric, to find the prototypes of the bow echo class, and then we use these prototypes in the classification step.

**A.3.1 Shape Matching with Skeleton Context**

In this section we want to introduce four indicators to measure the quality of the matched skeletons. The procedure for Skeleton matching is nearly the same with the method
introduced in [15], that is, instead of shape context, we use our proposed descriptor skeleton context. Defined in [15], we can compute the cost of mapping each skeleton point $P^1_i$ in image 1, to each skeleton point $P^2_j$ in image 2, which simply is the normalized difference between skeleton contexts of each pair of points in two images:

$$C(P^1_i, P^2_j) = \frac{1}{2} \sum_{k_1,k_2} \frac{(H_{SC}(P^1_i, r_{k_1}, \theta_{k_2}) - H_{SC}(P^2_j, r_{k_1}, \theta_{k_2}))^2}{H_{SC}(P^1_i, r_{k_1}, \theta_{k_2}) + H_{SC}(P^2_j, r_{k_1}, \theta_{k_2})}.$$ (A.13)

Having the cost of all possible mapping, we can use one of the algorithms designed to solve the bipartite graph matching, such as Hungarian method [123], which finds the minimum cost solution for matching points in image 1 to image 2. We define a threshold for the cost of matching, which indicates that if the minimum cost of matching one point from image 1 to the points of image 2 is greater than that threshold, then we announce that there is no matching point for this sample point in image 1. This would end in having some points without a proper pair from other image, and hence the matching ratio would be less than 1. This is necessary for having partial shape matching, which would be described in the next section. Next, we use an affine transformation to warp coordinates of one image, in order to map its skeleton to other image skeleton. As mentioned, for this step we could use TPS interpolation which tries to minimize the bending energy as it is defined in [15]. In warping coordinates, we need an indicator that shows the intensity of changes on shapes in the transformation. For instance, if the transformation is just a simple rotation or relocation, this indicator should be low, which means the transformation has not warped coordinates extensively. But it would be high, if the transformation is warping coordinates significantly to map the points together. If the affine transformation could be written in mathematical form as:

$$y = f(x) = Ax + b,$$ (A.14)

with matrix $A$ as linear map and vector $b$ as the offset for translation, then we can write our indicator for affine transformation as follows:

$$C_A = \log \frac{\sigma_1(A)}{\sigma_2(A)},$$ (A.15)

where $\sigma_i(A)$ shows $i$-th singular value of matrix $A$, with $\sigma_1 \geq \sigma_2$. The more $C_A$ is closer to zero, the more the skeleton of two shapes are similar, and the less coordinates are warped. Overall, we can use four different indicators for defining the distance between
two skeletons:

- **Matching Cost** ($C_{MC}$): This is defined based on the cost of matching points in two shapes, as it was computed in (A.13).

- **Bending Energy** ($C_{BE}$): The energy that TPS wants to minimize, which is described in [15].

- **Affine Indicator** ($C_A$): which is defined in (A.15).

- **Matching Ratio** ($C_{MR}$): It represents the ratio of the number of sample points in image 1, that we could match with points in image 2; to the total number of sample points. As it was discussed, some points would not match to any points if the cost of matching to the points in the other shape is greater than a predefined cost. This would allow the algorithm to achieve partial matching.

### A.3.2 Neighboring Effect

As it was mentioned, we need partial shape matching, as bow echoes have a tail in addition to the bow part in some cases, and this tail might be different among various bow echoes. Therefore, we should add an alteration to the shape matching part, in order to include partial shape matching in our algorithm. To do so, after the first iteration of shape matching, we can learn the initial mapping between sample points of two images. Because in our algorithm for extracting the edge list, points are listed in accordance with their spatial order, we can use this ordinal positions of sample points to define neighbors. For instance, if $(m-1)$-th and $(m+1)$-th sample points in image 1 are mapped respectively to $(n-1)$-th and $(n+1)$-th sample points in image 2, then we expect $m$-th sample point in image 1 to be mapped as close as possible to $n$-th sample point in image 2. This is what we call neighboring effect, and we want to impose this constraint in the shape matching algorithm by adding a Neighbor Cost to the cost introduced in (A.13). This Neighbor Cost for the point $m$ in the image 1 could be in the form of:

$$C_N(n, m) = \epsilon_N \left[ 1 - \exp \left( - \frac{(n - m - E\{\delta_N\})^2}{2\sigma^2} \right) \right], \quad (A.16)$$

where $\epsilon_N$ is the maximum amplitude of this cost, and random variable $\delta_N$ is the difference between sample points’ numbers in image 1 with respect to the their mapped sample points’ in image 2. This cost would be a function of indexes of sample points in image...
2 \textit{(i.e. n)} and image 1 \textit{(i.e. m)}, hence for each pair of \((m,n)\) we would calculate a neighboring cost according to this function to be added to their mapping cost. For instance, if the average of the mapping differences is 5, we expect to map the point 10 in image 2, to the point 5 in image 1. Based on (A.16), it would add 0 to the cost of mapping pair of \((10,5)\), however it increases as \(m\) deviating from 5. This Gaussian shape function would try to keep mapping of each sample point in accordance to its neighbors.

\section*{A.3.3 Distance Definition and Prototype Extraction}

Because there is no simple metric to distinguish between the shape of two different skeletons, defining the distance between them becomes more challenging. However, after introducing indicators for matching two skeletons in section A.3.1, we should be able to define a distance metric based on the combination of those indicators, and learn a classifier using that distance metric. In order to make this process more straightforward, we employ an iterative framework to define the distance metric and extract some prototypes of a bow echo class for the classification purposes, simultaneously.

Combining the skeleton matching indicators to define the distance metric, we use Principle Component Analysis (PCA) and this distance metric to find the best prototypes representing the bow echo class. Having abundant types of bow echoes in radar images, we can use \textit{K-medoids} algorithm \cite{86} to find prototypes in the bow echo class. Thereafter, these prototypes are used as the representatives of the bow echo class.

K-medoids algorithm after initialization and finding k initial medoids iterate between these two steps:

- **Assignment**: In this step we should partition the feature space based on the distance of the points to the medoids. Each point should be assigned to medoid \(m_i\) if and only if the distance of that point has the following property:

\[ D(x, m_i) \leq D(x, m_j), \quad \forall j \neq i, \quad (A.17) \]

where \(D(.,.)\) is the distance between two points, \(x\) is feature points in feature space, and \(m_j\)'s are medoids.

- **Update**: After assigning each point to a medoid, we can form clusters. Now we should find the point that has the minimum distance to all other points in a cluster. This point would be a new medoid.
As one can infer, to perform k-medoids algorithm, we need to have a well-defined distance metric between two points in the feature space (i.e., two skeletons in the real world). In Section A.3.1, in addition to the three matching costs showing the difficulty of that matching task, we define a matching ratio that shows how successful the task is. This parameter ranges from 0.5 to 1, where the greater the ratio the more successful the task. Hence, we reflect the extent of successfulness of the task on those three costs by inversely mapping the matching ratio from \([0.5, 1]\) to \([0, 1]\) and multiplying it to those costs. We use the exponential encoding function of \(e^{(0.5 - r) / 0.1}\) to approximately do the inverse mapping, which maps \(r = 0.5\) to 1 and \(r = 1\) to 0.0067.

After constructing the revised costs by reflecting the matching ratio on those three costs, now we can define our distance in this three dimensional feature space, by applying PCA, in order to reduce the dimension from three to one. Note that, applying PCA would reduce a high-dimensional feature space to a low-dimensional feature space with highest variation, and hence it would be helpful for classification and discrimination purposes. If we perform matching on all skeleton pairs available in the dataset, we would have \(r = N^2 / 2\) rows of matching costs in feature space (\(N\) is the total number of skeletons in the dataset). Therefore, the feature space would be a matrix \(C_{r \times 3}\), which has \(r\) samples and 3 dimensions. Each row in the feature matrix shows the matching costs of two skeletons in the dataset. What PCA is doing is simply as follow:

\[
t_{r \times 1} = C_{r \times 3} \cdot w_{3 \times 1},
\]

where \(w\) is the eigenvector associated with the highest eigenvalue of the matrix \(C^\top C\), and \(t\) is the reduced-dimension feature vector. Thus, the coefficients of \(w\) can be used to define the distance metric as follow:

\[
d(i, j, w) = [c'_{ij}, e'_{ij}, a'_{ij}] \cdot w,
\]

where \(c'_{ij}\), \(e'_{ij}\), and \(a'_{ij}\) are revised matching cost, bending energy, and affine indicator to match skeleton \(i\) to skeleton \(j\) respectively. Their definition is stated in the algorithm 7.

As we define the distance metric and the way to extract prototypes, we can make them happen at the same time in an iterative fashion. To do so, we design an algorithm, which iterates on the coefficients of the distance metric to extract prototypes and update the distance metric according to new prototypes, until it converges.

In a shape matching scenario, distance of a shape to a class would be its minimum distance to one of the class’s prototypes. Hence, if we have the prototypes for the bow
Figure A.10: Bow echo prototypes with $k = 8$. Each column shows a bow echo prototype with its extracted skeleton.

Echo class, we could reduce the number of rows in the feature matrix by just keeping the matching costs of each skeleton to its nearest prototype. This would reduce $r$ to $N$, which means we have one row for each skeleton in the dataset. However, this would be tricky, since by finding some new prototypes in a class, the feature matrix would change in accordance with the new prototypes. Hence, to rectify this issue and finding the optimum distance metric based on the prototypes, we design Algorithm 7. In each step, we first find the prototypes according to the distance metric defined in the last step from (A.19), and then with respect to the new prototypes update the feature matrix. Next, update the distance metric coefficients by applying PCA to the new feature matrix. It iterates until it converges to consistent coefficients. In the first step, all costs would contribute equally to the definition of the distance metric. The detailed algorithm is in 7. The output of this algorithm would be $k$ prototypes representing bow echo class, and the distance coefficients to map three-dimensional feature space into one dimension. For instance, when we use $k = 8$ on our dataset for year 2008, the resulting prototypes for the bow echo class are shown in Figure A.10.

A.3.4 Classifier

After defining prototypes for the bow echo class and distance coefficients, in order to decide whether a skeleton image belongs to bow echo class or non-bow echo class, we just need to find the minimum distance between skeleton image and bow echo prototypes. The distance of two skeletons is defined as a linear combination introduced in (A.19). As a classifier we use Mixture Discriminant Analysis (MDA) [53], which is generalized version of Linear Discriminant Analysis (LDA). In MDA, we consider each class has $R_m$ prototypes with Gaussian distributions, and next, using Expectation Maximization (EM) algorithm [34] to find the Gaussian-distributed parameters and probability of each sample.
Algorithm 7: Prototypes and Distance

Input: Matching Cost ($C_{MC} = [c_{ij}]$), Bending Energy ($C_{BE} = [e_{ij}]$), Affine Indicator ($C_{A} = [a_{ij}]$), and Matching Ratio ($C_{MR} = [r_{ij}]$), $w_0 = [0, 0, 0]^T$, $w_1 = [1, 1, 1]^T$, Skeletons = $[x_1, \ldots, x_N]$

function PrototypeExtraction(Skeletons):

$n = 1$
while $\|w_n - w_{n-1}\|^2 \geq \epsilon$ do

Prototypes $([m_1, \ldots, m_k])$ ← Randomly assign prototypes

while Prototypes change do

Assignment: For each point find the cluster index as follow:

$$j^* = \arg\min_j (d(x_i, m_j), w_n)$$

Update: Find new prototype in each cluster:

$$m'_j = \min_{i_j} \left( \sum_{l_j=1}^{L_j} d(x_{ij}, x_{lj}, b mw_n) \right)$$

end

FeatureSet ($C_{N \times 3}$) ← update feature set to have the matching costs of each skeleton to its nearest prototype

$n \leftarrow n + 1$

$w_n \leftarrow$ Coefficient of PCA on Feature Set

end

return Prototypes, $w_n$

in each Gaussian-distributed sub-class. Note that, these prototypes in EM algorithm are different from what we extracted in Section A.3.1. The feature that we are training the classifier on, is the distance of each skeleton from its nearest prototype defined in Algorithm 7. The EM algorithm alternates between these two steps [53]:

- **Expectation Step** (E-step): Given the parameters for the distributions of $R_m$ sub-classes in class $m$, we should assign a weight for each sample in each sub-class, with total sum of probabilities equal to one.

- **Maximization Step** (M-step): Using weights computed in the E-step, in order to compute weighted Maximum Likelihood estimation for parameters of each sub-class distribution.

After training and finding the parameters for sub-class distributions in both bow echo and non-bow echo classes, our classifier assess the weight of new coming samples for each
subclass in two classes. This would determine the class of that sample based on the label of the nearest subclass to the sample.

### A.4 Case Study

Using skeletonization approach and shape matching algorithm with skeleton context introduced in Sections A.2 and A.3 respectively, alongside with the classifier defined above, we are able to detect bow echoes in radar images automatically with high rate of accuracy. To test this approach and the classifier, we use our radar database for a case study.

#### A.4.1 Training Data

Our database consists of US radar images taken by NEXRAD radars in the whole year of 2008. We have chosen that year because it had a large number of severe weather activities. The techniques developed, however, are general and applicable to any year. We search the whole year images to find those dates that bow echo happened in United States, and succeed to label 89 distinct days with bow echo during 2008 including 1,148 radar images. After that, we extract skeleton of regions of interest in each radar images and label them as bow echo and non-bow echo classes. In each image that we can spot a bow echo, there would be some other parts that are not bow echo but would be captured as regions of interest, for their reflective signal amplitudes are similar to bow echo. From these images, we label 1,148 bow echo samples and 443 non-bow echo samples.

#### A.4.2 Prototype Extraction

As it was explained in section A.3.4, we need to implement k-medoids algorithm [87] to find \( k \) different bow echoes as prototypes of bow echo class. Consequently, we first find the matching costs between all bow echoes by running shape matching algorithm pairwise on all bow echo class samples. After finding matching costs matrix between all pairs of bow echoes, by running k-medoids algorithm, \( k \) bow echo prototypes are extracted, as it is depicted in Figure A.10, for \( k = 8 \).
A.4.3 Classification

The last step would be the training of the MDA classifier and classification process. For this step, we use cross-out validation to get more accurate results on our database. 20-fold cross-out validation would chunk data to 20 parts, each of which containing both bow echo and non-bow echo samples. In each iteration, one chunk of data would be considered as test data, and the other 19 chunks of data would be used as training data. The results of using MDA algorithm with 8, 9, and 10 prototypes \((k = 9, 8, 10)\) is shown in Figure A.11(b), which reveals that the best option for number of distributions is \(R_m = 2\). The effect of the number of bow echo prototypes is computed in Figure A.11(a). Results, indicate that overall it has increasing order with adding more prototypes, but it became approximately constant after 5 prototypes. Because for finding bow echo prototypes, the k-medoids algorithm should run separately each time, these prototypes could be completely different. For instance, for \(k = 3\) and \(k = 4\), the k-medoids algorithm can choose totally distinct prototypes. Thus, as our proposed algorithm depends on topological features of these distinct prototypes, the results of classification can vary based on these features. To compensate this effect, we apply moving average on the results, which makes it more reliable and robust against randomness inherent in choosing prototypes. We can go beyond ten prototypes to get better results, but the disadvantages of increasing computational time by adding more prototypes is higher than the benefits of small portion of improvement in accuracy rate.

Figure A.11: Accuracy Rate versus (a) number of bow echo prototypes, and (b) number of mixture distributions in each class.
The following tables show the meta-data related to the datasets that are being used in Chapters 5 and 6, and their sensitive features.

Table B.1: Adult train dataset with gender as sensitive feature

<table>
<thead>
<tr>
<th>Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>+1</td>
<td>1196</td>
</tr>
<tr>
<td>-1</td>
<td>9352</td>
</tr>
<tr>
<td>Total</td>
<td>10548</td>
</tr>
</tbody>
</table>

Table B.2: Adult test dataset with gender as sensitive feature

<table>
<thead>
<tr>
<th>Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Female</td>
<td>Male</td>
</tr>
<tr>
<td>+1</td>
<td>473</td>
</tr>
<tr>
<td>-1</td>
<td>3674</td>
</tr>
<tr>
<td>Total</td>
<td>4147</td>
</tr>
</tbody>
</table>

Table B.3: Adult train dataset with race sensitive feature. AIE: Amer-Indian-Eskimo, API: Asian-Pac-Islander

<table>
<thead>
<tr>
<th>Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIE</td>
<td>API</td>
</tr>
<tr>
<td>+1</td>
<td>37</td>
</tr>
<tr>
<td>-1</td>
<td>275</td>
</tr>
<tr>
<td>Total</td>
<td>312</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>8108</td>
</tr>
</tbody>
</table>
### Table B.4: Adult test dataset with race sensitive feature.

<table>
<thead>
<tr>
<th>Groups</th>
<th>AIE</th>
<th>API</th>
<th>Black</th>
<th>White</th>
<th>Other</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels</td>
<td>+1</td>
<td>16</td>
<td>104</td>
<td>132</td>
<td>21</td>
<td>3100</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>107</td>
<td>237</td>
<td>1049</td>
<td>86</td>
<td>9561</td>
</tr>
<tr>
<td>Total</td>
<td>123</td>
<td>341</td>
<td>1181</td>
<td>10909</td>
<td>107</td>
<td>12661</td>
</tr>
</tbody>
</table>

AIE: Amer-Indian-Eskimo, API: Asian-Pac-Islander

### Table B.5: COMPAS dataset with sex as sensitive feature.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Female</th>
<th>Male</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels</td>
<td>+1</td>
<td>422</td>
<td>1416</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>609</td>
<td>2831</td>
</tr>
<tr>
<td>Total</td>
<td>1031</td>
<td>4247</td>
<td>5278</td>
</tr>
</tbody>
</table>

### Table B.6: COMPASS dataset with race as sensitive feature.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels</td>
<td></td>
</tr>
<tr>
<td>+1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>


161


[115] Cecilia Munoz, Executive Office of the President, , Domestic Policy Council Director, Megan (US Chief Technology Officer Smith (Office of Science, Technology Policy)), DJ (Deputy Chief Technology Officer for Data Policy, Chief Data Scientist Patil (Office of Science, and Technology Policy)). Big data: A report on algorithmic systems, opportunity, and civil rights. Executive Office of the President, 2016.


Mohammad Mahdi Kamani

Vita

Education

2013 – 2015 **M.Sc. in Electrical Engineering, Communications**, *Sharif University of Technology.*
2008 – 2013 **B.Sc. in Electrical Engineering, Communications**, *Sharif University of Technology.*

Work Experiences

2019 **Machine Learning Research Intern**, *Twitter, Cortex.*
2018 **Machine Learning Research Intern**, *Twitter, Cortex.*
2017 **Machine Learning Research Intern**, *Honeywell International Inc.*

Selected Publications


Honors and Awards

2018 Awarded *AI for Earth* grant, Microsoft.
2017 Awarded the three-year *Academic Computing Fellowship*, Penn State University

✉️ mm.kamani7@gmail.com • 🌐 https://mmkamani.com