City Research Online

City St George's, University of London

Citation: Wang, R. (2025). Robust and fair learning: From correlation matrices to fair learning models. (Unpublished Doctoral thesis, City St George's, University of London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/35462/

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to City Research Online policy.

City Research Online: http://openaccess.city.ac.uk/ publications@citystgeorges.ac.uk

Robust and fair learning: From correlation matrices to fair learning models

Runshi Wang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

Bayes Business School.

Faculty of Actuarial Science and Insurance

Bayes Business School

City St George's, University of London

June 24, 2025

I, Runshi Wang, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

This thesis explores the use of convex optimisations to address two types of problems. In the first chapter, we review the nearest correlation matrix problem, a problem in actuarial science and finance, to find the nearest correlation matrix that is positive semidefinite. We introduce two algorithms to solve the problem, namely the iterative quadratic/linear programming method and the gradient descent method. The iterative quadratic/linear programming method enjoys great flexibility so that it can handle different types of norms and user-defined constraints. The gradient descent method works for unconstrained problems under the Frobenius norm and experiments show that it is resilient to noise. In the second chapter, we explore the fair learning problem. We introduce different definitions of fairness in classification tasks and how they can be generalised into regressions. We propose two fair regression models based on Liu-type estimator, using the expected squared difference of the pairwise linear components and coefficient of determination of the sensitive features as measures of fairness, respectively. The first method works with a single sensitive feature while the second method can include multidimensional sensitive features into its model. Both models can be calculated as closed-form solutions. In the third chapter, we continue our study of the fair learning problem and propose a fair generalised linear model framework that uses the maximum mean discrepancy as the fairness measure. Our choice of fairness measure can capture more complex differences between distributions from different sensitive groups. The model can be applied to datasets with different outcome types so is suitable for both classification and regression tasks.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Prof. Vali Asimit and Dr. Rui Zhu, for their invaluable guidance, patience, encouragement, and support throughout this journey.

I am also deeply grateful to Dr. Dimitrina Dimitrova for supporting me, especially during the challenging times of the Covid pandemic.

I would also like to express my thanks to Bayes Business School. I am grateful to all my colleagues and friends, and all the faculty members who have made this journey enjoyable.

Finally, I want to express my deep gratitude to my family, my father, Jingyu Wang, and my mother, Li Zhao, for their encouragement and unconditional love.

Contents

Introduction **15**

1 Efficient Positive Semidefinite Matrix Approximation by Iterative Optimisations and Gradient Descent Method **19**

1.1 Introduction 19

1.2 Problem Formulation 22

1.3 Related Work 24

1.4 Proposed PSDisation Methods 27

1.4.1 Iterative Quadratic/Linear Programming 27

1.4.2 Gradient Descent 33

1.5 Experimental Results on Simulated Data 34

1.5.1 Experiments on Iterative Quadratic/Linear Programming Algorithm with Simulated Data 36

1.5.2 Experiments on Gradient Descent Method with Simulated Data 40

1.5.3 Experiments with Simulated Data Rounded to Multiples of 25% 43

1.6 Experiments Results on Real Datasets 45

1.6.1 Experiments with Assets Daily Return Data 45

1.6.2 Experiments with Machine Learning Data for Applications in PCA 49

1.7 Conclusions 54

2 Two Fair Regression Models with Liu-type Estimator 56

2.1 Introduction . . . 56

2.2 Related Work . . . 59

2.2.1 Fairness in Classification Tasks . . . 59

2.2.2 Fairness in Regression Tasks . . . 62

2.3 Methodology . . . 66

2.3.1 Liu-type Estimator . . . 66

2.3.2 Fair Liu Model . . . 68

2.3.3 Fair Liu Model with Sensitive Features . . . 69

2.4 Experimental Results . . . 72

2.4.1 Experiments on DAG Simulated Data . . . 72

2.4.2 Experiments on Real Dataset . . . 78

2.4.3 Tests of Stability . . . 82

2.4.4 Cross-Validation . . . 85

2.5 Conclusions . . . 88

3 Fair Generalised Linear Model with the Maximum Mean Discrepancy Penalty 90

3.1 Introduction . . . 90

3.2 Related Work . . . 92

3.3 Methodology . . . 94

3.3.1 Fair Generalised Linear Model with a Convex Penalty . . . 94

3.3.2 Maximum Mean Discrepancy . . . 96

3.3.3 Fair Generalised Linear Model with the Maximum Mean Discrepancy Penalty . . . 97

3.4 Experimental Results . . . 99

3.5 Conclusions . . . 107

Appendices 108

A Additional Experimental Results for Fair Liu Model and Fair Liu Model with Sensitive Features 108

Bibliography **115**

List of Figures

1.1 Flowchart of Algorithm 1: An iterative quadratic/linear programming algorithm to solve PSDisation under some matrix norm 31

1.2 Flowchart of Algorithm 2: The Gradient descent algorithm for PSDisation under the F-norm 35

1.3 Percentage of best performances by ALD, GD and IQP versus standard deviation of the noise in PSDisation tests with different dimensions of the initial matrix **A**. 42

1.4 Cumulative sum of the variance of each principal component in the transformed wine data. 51

2.1 DAG used to simulate the dataset. 72

2.2 Experimental results on the DAG simulated dataset generated with different h values for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$) on training (left) and testing (right) sets of different linear models fitted without sensitive features. Both values are the lower the better. 75

2.3 Experimental results on the DAG simulated dataset generated with different h values for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{EO}$) on training (left) and testing (right) sets of different linear models fitted without sensitive features. Both values are the lower the better. 76

2.4 Experimental results on the DAG simulated dataset generated with different h values for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by CoD) on training (left) and testing (right) sets of different linear models fitted with sensitive features. Both values are the lower the better. 77

2.5 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$) on training (left) and testing (right) sets of the lsac dataset with black versus non-black race as sensitive feature of FGLM and fLiu. 80

2.6 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{EO}$) on training (left) and testing (right) sets of the lsac dataset with black versus non-black race as sensitive feature of FGLM and fLiu. 80

2.7 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by CoD) on training (left) and testing (right) sets of the lsac dataset with black versus non-black race as sensitive feature of FRRM and fLiuS. 81

3.1 Average experimental results for trade-offs of prediction accuracy and fairness on training sets of the lsac dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by RMSE, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Middle rows and bottom left: MMD with different values of σ. Bottom right: KS statistic. 102

3.2 Average experimental results for trade-offs of prediction accuracy and fairness on testing sets of the lsac dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by RMSE, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Middle rows and bottom left: MMD with different values of σ. Bottom right: KS statistic. 103

3.3 Average experimental results for trade-offs of prediction accuracy and fairness on training sets of the German credit dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by predictive error rate, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Bottom left: MMD with $\sigma = 1$. Bottom right: MMD with $\sigma = 10$. 105

3.4 Average experimental results for trade-offs of prediction accuracy and fairness on testing sets of the German credit dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by predictive error rate, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Bottom left: MMD with $\sigma = 1$. Bottom right: MMD with $\sigma = 10$. 106

A.1 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the standardised DAG simulated dataset of FRRM, fLiu, FGLM and fLiuS. 110

A.2 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the lsac dataset (standardised) with races (Asian, black, Hispanic, white and other) as sensitive feature of FRRM, fLiu, FGLM and fLiuS. 111

A.3 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the student_performance dataset (standardised) with black versus non-black race as sensitive feature of FRRM, fLiu, FGLM and fLiuS. 112

A.4 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the crime dataset (standardised) with black versus non-black race as sensitive feature of FRRM, fLiu, FGLM and fLiuS. 113

A.5 Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the parkinsons_updrs dataset (standardised) with black versus non-black race as sensitive feature of FRRM, fLiu, FGLM and fLiuS. 114

List of Tables

1.1 The F-norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}$ obtained from different algorithms on different dimensions of the initial matrix **A**. (Best performances are marked in bold.) 36

1.2 The Chebyshev norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}$ obtained from different algorithms on different dimensions of the initial matrix **A**. (Best performances are marked in bold.) 37

1.3 Relative change in the F-norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}$ compared with the Newton method in percentage. (Best performances are marked in bold.) 37

1.4 Relative change in the Chebyshev norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}$ compared with the Newton method in percentage. (Best performances are marked in bold.) 37

1.5 Running time of different algorithms on different dimensions of the initial matrix **A** (in seconds). (Best performances are marked in bold.) 38

1.6 Optimal solution by ALD and IQP for constrained and unconstrained PSDisation problems under the F-norm and the H-norm on different dimensions of the initial matrix **A**. (Best performances are marked in bold.) 39

1.7 Convergence time and optimal solution by ALD, IQP, GD and SDM for unconstrained PSDisation problems under the F-norm on different dimensions of the initial matrix **A**. (Best performances are marked in bold.) 40

1.8 Average Frobenius distance from $\mathbf{X}$ to $\mathbf{A}$ and $\mathbf{A}'$ by different algorithms over 100 tests in experiment settings. (Best performances are marked in bold.) 41

1.9 Average Frobenius distance from $\mathbf{X}$ to $\mathbf{A}'_s$ and $\mathbf{A}'_s$ by different algorithms over 500 tests in experiment settings. (Best performances are marked in bold.) 44

1.10 Average Frobenius distance from $\mathbf{X}$ to $\mathbf{A}'_t$ and $\mathbf{A}'_t$ by different algorithms over 500 tests in experiment settings. (Best performances are marked in bold.) 44

1.11 Frobenius distance and Chebyshev distance between $\mathbf{A}$ and $\mathbf{X}$ by different algorithms on NASDAQ stocks return correlation matrix. (Best performances are marked in bold.) 46

1.12 Frobenius distance and Chebyshev distance between $\mathbf{A}$ and $\mathbf{X}$ by different algorithms on a mixture of NASDAQ and SP500 stocks return correlation matrix. (Best performances are marked in bold.) . 47

1.13 Annualised average return, annualised standard deviation and Sharpe ratio of NASDAQ stocks minimum variance portfolios based on correlation matrices estimated using different methods. (Best performances are marked in bold.) 48

1.14 Annualised average return, annualised standard deviation and Sharpe ratio of a mixture of NASDAQ and SP500 stocks minimum variance portfolios based on correlation matrices estimated using different methods. (Best performances are marked in bold.) . 48

1.15 Variance of each principal component in the transformed wine data. (Higher variance in each row is marked in bold.) 50

1.16 Weight of each feature assigned by the 1_{st} and 2_{nd} principal components in the wine data by PCA, pairwise PCA and all PCA. Top: Weights of features in first principal component. Bottom: Weights of features in second principal component. (Top 3 weights in each column are marked in bold.) 52

1.17 Average testing accuracy of kNN models built on data after PCA using different correlation matrix estimating methods. (Best performance is marked in bold.) 54

2.1 Comparisons of two definitions for regression fairness. 66
2.2 Summary and comparisons between FGLM, fLiu, FRRM and fLiuS. 71
2.3 Average computational time of LM, FGLM, FRRM, fLiu and fLiuS on different datasets (in second). (Lowest computational time in each column is marked in bold.) 82
2.4 Stability, accuracy and fairness measured of fLiuS and FRRM models with different parameters on the lsac dataset with noise of standard deviation 0.2. 83
2.5 Stability, accuracy and fairness measured of fLiuS and FRRM models with different parameters on the lsac dataset with noise of standard deviation 1. 84
2.6 Stability, accuracy and fairness measured of fLiuS and FRRM models with different parameters on the lsac dataset with noise of standard deviation 0.02. 84
2.7 Stability, accuracy and fairness measured of LM, fLiu, fLiuS (with cross-validation k) and FRRM models with different parameters on the lsac training dataset with noise of standard deviation 0.2. 86
2.8 Stability, accuracy and fairness measured of LM, fLiu, fLiuS (with cross-validation k) and FRRM models with different parameters on the lsac testing dataset with noise of standard deviation 0.2. 87

Introduction

A correlation matrix plays an important role in various applications across finance, risk management, machine learning, data science, engineering, and signal processing [1, 2, 3, 4]. It quantifies pairwise relationships between variables. A valid correlation matrix should be symmetric, positive semidefinite with unit diagonal entries. However, in real-world applications, an empirically computed correlation matrix often fails to satisfy the positive semidefinite requirement, due to missing data, estimation errors, numerical instability, or rounding errors [5]. Therefore, an algorithm is needed to find a valid positive semidefinite correlation matrix which is nearest to the estimated one. This problem is called PSDisation.

One fundamental question to consider before solving the PSDisation problem is how we define nearness. This is usually done by choosing a matrix norm. [6] is one of the earliest works on this topic which uses the Frobenius norm. Many recent literatures use the same definition [7], while some can be adapted to work with the W-norm or H-norm [8, 9]. However, it is suggested that alternative norms such as the Chebyshev norm can be considered according to different application scenarios [5].

In Chapter 1, we review the choices of norms for the PSDisation problem and some state-of-the-art methods in the current literature. We then propose two new algorithms as solutions to this problem. Our contribution is to provide an efficient way to find the nearest correlation matrix which is more flexible with the choice of norms, and another method which is robust so that it behaves stable on data with noise.

With the fast development of machine learning techniques, biases have become

a huge issue, leading to unfair decisions over different demographic groups. As a result, fair learning became a crucial research topic in recent decades. It aims to adjust the learning model, allowing some compromise on prediction accuracy, so that it does not systematically favour or disadvantage specific sensitive groups such as race, gender, or age. For example, we may expect that males and females with similar backgrounds have equal opportunities to get a job, or that a person is not rejected for loans simply because of ethnicity group.

Different measures of fairness have been defined in order to satisfy the requirements of specific applications and fair learning algorithms have been proposed accordingly [10]. Most early fair learning studies focus on classification tasks [11, 12, 13, 14, 15], while [16, 17, 18, 19] apply to fair regression. [20] proposed a fair learning framework based on generalised linear models which can be flexibly applied to data with different types of outcomes. However, the fairness defined in their model only captures the first two moments of the distributions of predicted outcomes and lacks robustness.

In Chapter 2, we introduce the fairness problem in learning. Starting from classification tasks, we summarise how fairness can be defined and how it can be adapted to regression models. We proposed two fair regression models based on Liu-type estimator. We show that our two models can be fitted to optimise fairness with a closed-form solution. Experiments show that they can perform as well as state-of-the-art methods while being more resilient to noise.

Chapter 3 explores the topic of achieving fairness in both classification and regression tasks. We propose to use a new measure of fairness, maximum mean discrepancy, as the penalty term to build our fair generalised linear model. We test our proposed method with datasets for both regression and binary classification tasks and show that it gives better fairness-accuracy trade-offs with certain measures of fairness.

Chapter 1 - Efficient Positive Semidefinite Matrix Approximation by

Iterative Optimisations and Gradient Descent Method

— This chapter was published in [21] as a feature article and was selected to be the cover of that issue.

We devise two algorithms for approximating solutions of PSDisation, a problem in actuarial science and finance, to find the nearest valid correlation matrix that is positive semidefinite. The first method converts the PSDisation problem with a positive semidefinite constraint and other linear constraints into iterative Linear Programmings (LPs) or Quadratic Programmings (QPs). The LPs or QPs in our formulation give an upper bound of the optimal solution of the original problem, which can be improved during each iteration. The biggest advantage of this iterative method is its great flexibility when working with different choices of norms or with user-defined constraints. Second, a gradient descent method is designed specifically for PSDisation under the Frobenius norm to measure how close the two matrices are. Experiments on randomly generated data show that this method enjoys better resilience to noise while maintaining good accuracy. For example, in our experiments with noised data, the iterative quadratic programming algorithm performs best in more than 41% to 67% of the samples when the standard deviation of noise is 0.02, and the gradient descent method performs best in more than 70% of the samples when the standard deviation of noise is 0.2. Examples of applications in finance, as well as in the machine learning field, are given. Computational results are presented followed by discussion on future improvements.

Chapter 2 - Two Fair Regression Models with Liu-type Estimator

We introduce two robust fair regression models which are simple to learn and have stable performance on dataset with noise. Both models are based on Liu-type estimator, a generalised version of Liu estimator with two parameters, the shrinkage parameter k and the correction parameter d. The first method considers the expected squared difference of the pairwise linear components as its fairness measure. It does not include sensitive features in its model prediction and d can

be calculated as a closed-form solution to minimise fairness when k is fixed. By alternating k and d we get a trade-off between fairness and predictive accuracy. The second method is constructed with a similar idea. However, it differs from the first method in two ways. Firstly, it includes multi-dimensional sensitive features in the model and, secondly, it uses coefficient of determination of the sensitive features as the measure of fairness. Our two fair regression models enjoy great simplicity. We use a simple simulated dataset as well as real datasets to show the performance of them on fairness-accuracy trade-off comparing with state-of-the-art methods. Furthermore, we test the stability of the models and find that our models are robust when data come with added noise.

Chapter 3 - Fair Generalised Linear Model with the Maximum Mean Discrepancy Penalty

We propose a new fair learning model based on generalised linear models for both classification and regression tasks. Fairness is achieved by introducing a penalty term defined by maximum mean discrepancy. This is a kernel-based measure that can effectively capture differences between distributions in high order. Both log-likelihood term of the generalised linear model and the penalty term for fairness are convex and thus the loss function can be optimised efficiently. The flexibility of generalised linear models ensures that the model can be applied to data with different types of outcomes, including continuous, binary, count, and multi-class, etc. We use real datasets with continuous and binary outcomes to compare its performance with the fair generalised linear models which uses expected squared difference of the pairwise linear components as fairness measure. Experimental results suggest that our fair learning model provides better fairness-accuracy trade-off when fairness is measured by maximum mean discrepancy or sum of squared differences of the pairwise expected outcomes.

Chapter 1

Efficient Positive Semidefinite Matrix Approximation by Iterative Optimisations and Gradient Descent Method

1.1 Introduction

A correlation matrix is a square matrix summarising correlation coefficients between each pair of variables. This is used by many financial or insurance companies to determine how a group of risks are dependent from each other [22]. It is useful across various fields to help us understand the relationship between variables. For example, in finance, a correlation matrix is used in optimisation to build a portfolio to minimise the volatility. It also enables efficient computation of cardinality constrained efficient frontiers to solve mean-variance portfolio selection problems [4]. In actuarial science, insurance companies use correlation matrices to measure insurers' exposure to risks and calculate the Solvency Capital Requirement (SCR) under the Solvency II Standard Formula [1]. Another application is to work with copulas for aggregation of risks in a more complex capital model [1]. In genomics science, correlation matrices help identify relationship between genes and construct relevance networks and association networks [2]. They also play a vital role in

genes classifications. In machine learning and data science, correlation matrices are widely used to identify key patterns of data through dimensionality reduction techniques such as principal component analysis (PCA) [3].

In all these applications, the correlation matrices are required to be symmetric and positive semidefinite (PSD) with diagonal entries all equal to one. In practice, however, the correlation matrix estimated from the empirical data is rarely PSD due to a variety of reasons, including data incompleteness, noise, rounding, manual adjustment or inconsistent computing approaches, etc [5]. For example, when correlations between variables are estimated in groups and then joined together, the resulting correlation matrix can be non-PSD. There are many other factors that contribute to this problem. The data collected in many scenarios can be incomplete and may need further repair. The correlations may be estimated using inconsistent approaches or have been manually adjusted according to the needs. All of these reasons lead to an invalid estimation of the correlation matrix [5]. Therefore, a preprocessing method is needed to obtain a valid PSD correlation matrix based on the empirical matrix in order to perform any further analysis. The problem of finding such correlation matrix nearest to an empirical matrix is called PSDisation [1].

Currently most state-of-the-art PSDisation methods focus on finding the nearest correlation matrix with respect to the F-norm, of which some apply to H-norm as well. Over those algorithms, the Newton method (NM) [8] and the augmented Lagrangian method (ALD) [9] are the most efficient in producing optimal solutions. However, depending on the industrial situations, insurance companies may wish to measure the similarity using an alternative norm other than the F-norm, which makes some algorithms more time-consuming or invalid. Another PSDisation method that is widely used in industry is the shrinking method [23]. It is a simple and efficient method that can improve the result of the alternating projections algorithm (APM) [5]. However, a target matrix that lies inside the cone of PSD matrices is needed for the shrinking method, and therefore its performance depends heavily on the choice of the target matrix.

[24] introduced a shrinkage method in order to improve the stability of the esti-

mated covariance matrix that is not positive definite or is ill-conditioned. This often happens when the matrix size is large and the number of observations is smaller than or comparable to the number of variables. It achieves its goal by reducing variance of the estimator, and thus reducing the mean squared error (MSE) via a computationally inexpensive approach. This involves a shrinkage of the sample covariance matrix (unbiased) towards a target matrix without variance, usually an identity matrix. The optimal shrinkage intensity parameter can be computed analytically to minimise the MSE. The similar idea can be used to obtain a PSD correlation matrix starting from an invalid one. Although the shrinkage method is not defined as a PSDisation method, it has been widely used in many applications in different fields. We will include this method in our experiments on real datasets to compare its performance with other PSDisation methods.

Therefore, most algorithms in the current literature primarily operate under the F-norm or the H-norm for the general PSDisation problems. However, they tend to lack flexibility when addressing more specific needs in specialised cases. To tackle this limitation, we propose two new methods to approximate the solution of the PSDisation problem, which offer greater flexibility and provide robust solutions to practical real-world challenges. The first algorithm works by solving a series of optimisations, such as linear programming (LP) or quadratic programming (QP). Instead of solving the original problem which requires the correlation matrix $\mathbf{X}$ to be PSD, our formulation requires $\mathbf{X} = \mathbf{U}^T\mathbf{Q}\mathbf{U}$ where $\mathbf{Q}$ belongs to the convex cone of diagonally dominant symmetric matrices with non-negative diagonal entries and thus all constraints are linear. This is a rich subset of the PSD matrices set, but the optimisations can be solved more efficiently due to the linearity of the constraints. The main advantage of this method is its flexibility as our formulation is adapted directly from the original problem and is thus effective on any choice of norms, including the F-norm, the Chebyshev norm, and the H-norm, or potentially a combination of any of these norms above. The advantages of using these norms will be illustrated in the results of our experiments. The second algorithm is to find the nearest correlation matrix to the initial matrix using an iterative gradient projection

method under the F-norm. This can be done simply by repeatedly taking a step along the gradient of the objective function and then projecting the matrix back to the PSD cone. This algorithm is easy to implement and enjoys good efficiency while maintaining good accuracy.

We will compare our two proposed algorithms with the state-of-the-art NM, ALD, and the shrinkage method. Extensive experiments demonstrate that our first iterative algorithm could achieve results comparable to state-of-the-art methods, while showing more flexibility to handle complex constraints or different choices of norms. Our second gradient descent method provides slightly less accurate results, but runs faster than the first algorithm and is more resilient to noise than some state-of-the-art methods and therefore can be more reliable in practical situations.

This chapter is organised as follows. In Section 1.2 we introduce the notations and formally define the PSDisation problem. Section 1.3 summaries some of the most popular PSDisation algorithms that exist in the current literature together with the shrinkage method by [24]. In Section 1.4 we introduce our two approaches to the PSDisation problem. Experiment results with simulated data as well as real data for both algorithms are reported in Section 1.5 and Section 1.6, respectively. Finally, in Section 1.7 we conclude this chapter and propose some future work.

1.2 Problem Formulation

We now introduce the notations we use throughout this chapter and the background knowledge of the PSDisation problem. Denote the set of $n \times n$ real matrices by $\mathbb{R}^{n\times n}$. A real-valued symmetric matrix $\mathbf{Z} \in \mathbb{R}^{n\times n}$ is positive definite (PD) if $\boldsymbol{y}^T\mathbf{Z}\boldsymbol{y}$ is positive for any nonzero real vector $\boldsymbol{y} \in \mathbb{R}^n$. It is positive semidefinite (PSD) if $\boldsymbol{y}^T\mathbf{Z}\boldsymbol{y}$ is non-negative for any nonzero real vector $\boldsymbol{y}$. Equivalently, a real symmetric matrix is PSD if all its eigenvalues are non-negative. We write $\mathbf{Z} \succeq 0$ if $\mathbf{Z}$ is PSD ($\mathbf{Z} \succ 0$ for PD). We use $\mathrm{diag}(\mathbf{Z})$ to denote the vector of all diagonal entries of $\mathbf{Z}$. Let $\mathbf{I}$ be the identity matrix and $\mathbf{1}$ be the all-ones vector. The operator $||\cdot||$ over the set of real matrices represents a generic matrix norm, for which different choices are signified by a subscript and four choices are explored in this chapter; namely,

for a $\mathbf{M} \in \mathbb{R}^{n\times n}$, these norms are as follows:

1. *Frobenious norm (or F-norm)*: $||\mathbf{M}||_{\mathbf{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^2} = \sqrt{\mathrm{Tr}(\mathbf{M}^{\mathbf{T}}\mathbf{M})}$;
2. *W-norm*: $||\mathbf{M}||_{\mathbf{W}} = ||\mathbf{W}^{\frac{1}{2}}\mathbf{M}\mathbf{W}^{\frac{1}{2}}||_{\mathbf{F}}$, where $\mathbf{W} \in \mathbb{R}^{n\times n}$ is a square matrix with positive entries;
3. *H-norm*: $||\mathbf{M}||_{\mathbf{H}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}m_{ij}^2}$;
4. *Chebyshev norm (or max norm)*: $||\mathbf{M}||_{\mathbf{C}} = \max_{i,j}(|m_{ij}|)$,

where m_{ij} is the (i, j) entry of $\mathbf{M}$.

The PSDisation problem we consider in this chapter is defined as follows. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric matrix that is not PSD with diagonal entries all equal to 1. It represents an empirical correlation matrix of n random variables $Y_1, \ldots, Y_n$ with standard deviations $\sigma_{Y_1}, \ldots, \sigma_{Y_n}$ whose (i, j) entry is given by

$$a_{ij} = \mathrm{corr}(Y_i, Y_j) = \frac{\mathrm{cov}(Y_i, Y_j)}{\sigma_{Y_i}\sigma_{Y_j}}, \text{ if } \sigma_{Y_i}\sigma_{Y_j} > 0.$$

We aim to find the nearest PSD matrix $\mathbf{X}$ of the same form. The problem can be written as follows.

$$\min_{\mathbf{X}} ||\mathbf{A} - \mathbf{X}||^2 \quad s.t.\ \mathbf{X} \succeq 0, \quad \mathrm{diag}(\mathbf{X}) = \mathbf{1}. \tag{1.1}$$

As we wish to find the nearest correlation matrix $\mathbf{X}$ to the empirical matrix $\mathbf{A}$, the choice of the matrix norm $||\cdot||$ determines how "nearest" is defined.

The F-norm [25] is a trivial choice for defining how close two matrices are and is widely used in most of the PSDisation literature. However, recent studies have explored the use of different norms in order to meet more practical demands. The W-norm is a weighted norm commonly used in numerical mathematics [6] and allows us to force some elements of $\mathbf{X}$ to be closer to the corresponding entries in $\mathbf{A}$. Setting $\mathbf{W}$ to the identity matrix will retrieve the F-norm. However, when applied to actuarial science, an insurance company may be more interested in the H-norm [1]. The H-norm is similar to the F-norm but allows one to assign weights to $\mathbf{M}$

on an element-by-element basis. Therefore, in the situation where one has prior knowledge on the correlation matrix, the H-norm may be preferred. For example, when one is more confident in some entries of the estimated correlation matrix **A** and want to fix them, he can allocate more weights on those entries so that a small change results in a huge penalty when measuring the distance between matrices. The H-norm is equivalent to the F-norm when **H** is an all-ones matrix. Finally, the Chebyshev norm specifies an element-wise ceiling for the largest difference between the entries of **X** and **A**, which can be particularly useful and more robust to outliers when m_{ij} is large.

It is worth mentioning that the choice of norms should be determined according to specific applications and there is no guarantee which one would work best. Therefore, we attempt to develop a more flexible algorithm that can be generalised to work with different norm choices.

1.3 Related Work

This section summaries some most popular existing solutions on PSDisation as well as the shrinkage method to fix the ill-conditioned covariance matrix.

One of the early algorithms to tackle the PSDisation problem is the alternating projections method (APM) [6]. It finds the nearest PSD correlation matrix by iterative projections onto two convex sets, the PSD matrices cone $\mathscr{S}$ and the set of matrices $\mathscr{U}$ with diagonal values all equal to one. The alternating projections method stated that by repeatedly projecting **A** onto $\mathscr{S}$ and then onto $\mathscr{U}$ we could finally obtain the nearest matrix **X** on the intersection of $\mathscr{S}$ and $\mathscr{U}$ [26]. Furthermore, to achieve the convergence to the optimal solution, a correction needs to be made according to [27]. In general, the alternating projections algorithm converges linearly at best [28] and can be applied under the F-norm as well as the W-norm. It shows good potential on the H-norm if an efficient algorithm of projection onto $\mathscr{S}$ can be found.

[6] suggested another way to tackle this problem, that is, to formulate the optimisation as a semidefinite programming (SDP) which can then be solved using

any powerful SDP solver. For the F-norm, the object $||\mathbf{A}-\mathbf{X}||^2$ is a second-order function of $\mathbf{X}$, so a transformation into the standard SDP primal form is needed before any general SDP solver can be applied. According to [6], the problem can be written as a standard SDP with $(n^2+n+2)^2$ variables and $\frac{1}{2}n^4+\frac{1}{2}3n^2+n+1$ constraints. Although the SDP problem can be further reformulated and simplified, to our knowledge, there is still currently no SDP solver that is efficient enough to solve such an SDP easily.

[7] introduced a method to produce a feasible correlation matrix near $\mathbf{A}$ based on decomposition of a valid correlation matrix by representing it using angular parameters on a multidimensional unit hypersphere. The constraints on $\mathbf{X}$ are therefore automatically satisfied. Then by solving an optimisation problem over the parameters we expect to recover a feasible $\mathbf{X}$ near $\mathbf{A}$ under a norm of our choice. One advantage of the hypersphere decomposition method (HDM) is that this algorithm allows us to perform PSDisation with the H-norm, so we are able to weight each entry with more flexibility. Furthermore, the computation is efficient and the converge speed is fast on small matrices. However, according to [1], the hypersphere decomposition method may suffer from local optimum, making the solution less accurate. Several side effects are discussed in their paper. For instance, when changing the order of the input variables, the output of this method differs, which should not happen to any proper PSDisation algorithms theoretically.

The spectral decomposition method (SDM) [7] is a slightly rough approach but can give an acceptable solution even faster. The idea is to set all negative eigenvalues to zero after performing a spectral decomposition of the initial matrix. A rescaling is then carried out to ensure the resulting correlation matrix has diagonal elements all equal to one. The procedure is extremely fast as it contains no iterations. The solution found by the spectral decomposition method can be an approximation of the true optimal of the PSDisation problem, or can be used as a starting point for other PSDisation methods [7]. However, the result cannot be considered as accurate and it lacks the flexibility to be adapt with different choice of norms.

[8] introduced a Newton-type algorithm to find the nearest correlation matrix

under the F-norm. It is designed to work with the F-norm as well as the W-norm using the semismooth Newton method combined with the conjugate gradient solver. It has a quadratic convergence rate, which is superior to the APM, and is effective on data with thousands of dimensions. Further improvements in this method were suggested by [29]. [9] extended their work to the H-norm case based on an augmented Lagrangian dual approach (ALD) by avoiding computing the projection onto the PSD cone under the H-norm. An upper bound or lower bound for each entry of the correlation matrix can also be set.

[24] proposed a shrinkage method for estimating the covariance matrix in order to fix the ill-conditioned sample covariance matrix calculated from observations in high-dimensional settings, which is likely to occur when the number of samples available is smaller than the number of variables. It achieves higher accuracy and better stability by shrinking the sample covariance matrix towards a simple structured target matrix. This significantly reduces the variance of the estimator without introducing too much bias. The degree of shrinkage is called shrinkage intensity and can be optimised to minimise the MSE in a closed form. The shrinkage method has been particularly useful in finance applications such as portfolio optimisation or risk management. It is also commonly adopted in statistics or and machine learning fields such as genomics and bioinformatic analysis. [23] developed a shrinking method using a similar idea for restoring positive semidefiniteness. It was designed to handle correlation matrix where one wants to keep some diagonal blocks fixed. Instead of minimising MSE using the observation data, it uses a bisection method to find the optimal solution to the PSDisation problem that lies on the path between the initial non-PSD matrix $\mathbf{A}$ and a target PSD matrix. The bisection method enjoys great simplicity and requires few iterations before achieving a good tolerance. However, the outcome of this method depends heavily on the choice of the target matrix. Indeed, the target matrix can be chosen as any PSD matrix that is near to $\mathbf{A}$. Therefore, the shrinking method is not considered suitable for finding the global optimal of the PSDisation problem. [5] explored the shrinking method as a way to improve the results of the alternating projections method. Furthermore, any block-structured

constraints that are linear can be applied in this method. We will include the more general and widely used shrinkage method by [24] in our experiments to see how our PSDisation algorithms and other state-of-the-art methods perform on real data applications in finance and machine learning comparing with this technique.

Overall, most algorithms may become computationally expensive when additional constraints are introduced to the problem. To the best of our knowledge, there is currently no algorithm that is able to solve the general PSDisation problem under the Chebyshev norm efficiently.

1.4 Proposed PSDisation Methods

In this section, we propose two methods to approximate the nearest correlation matrix. Currently most PSDisation methods are designed specifically for the F-norm. Some of them can be adapted to work with the H-norm with the cost of a more complex formulation or a longer computational time. We want to produce an algorithm that is more flexible with the objective and the constraints and more efficient than traditional SDP solvers. Therefore we propose our first proposed method which is done by iteratively solving a series of linearly constrained quadratic optimisations or linear optimisations. We get an approximation of the optimal value of the PSDisation problem by directly minimising the objective function at each iteration, where we have the flexibility to easily choose different norms to work with. The other approach we propose is based on iteratively taking a step along the gradient to reduce the objective function and then projecting the resulting matrix onto the PSD matrices set with diagonal one according to the spectral decomposition method [7], which is designed for solving the PSDisation problem with respect to the F-norm. Experiments show that our gradient descent approach enjoys good efficiency and robustness. MATLAB implements for both algorithms are made available online.

1.4.1 Iterative Quadratic/Linear Programming

Consider the PSDisation problem in (1.1), which requires $\mathbf{X}$ to be PSD in its constraints. The PSD constraint can be reduced and therefore the optimisation problem is transferred into quadratic or linear optimisation problems with linear constraints.

A square matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ is called diagonally dominant if

$$|x_{ii}| \geq \sum_{j \neq i} |x_{ij}| \quad \forall\, 1 \leq i \leq n,$$

where x_{ij} is the (i, j) entry of $\mathbf{X}$. It is well known that a symmetric diagonally dominant matrix with non-negative diagonal entries is PSD, which makes up a rich subset of PSD matrices. Therefore, we could approximate the optimisation problem (1.1) stated above by solving convex optimisation problems of the following form in an iterative manner,

$$\begin{aligned} &\min_{\mathbf{X},\mathbf{Q}} \; ||\mathbf{A} - \mathbf{X}||^2 \\ &s.t. \;\; \text{diag}(\mathbf{X}) = \mathbf{1}, \\ &\quad \mathbf{X} = \mathbf{U}_h^T \mathbf{Q} \mathbf{U}_h, \\ &\quad \mathbf{Q} = \mathbf{Q}^T, \\ &\quad \mathbf{Q} \text{ is diagonally dominant with non-negative diagonal.} \end{aligned} \tag{1.2}$$

Note that $\mathbf{U}_1 = \mathbf{I}$ and $\mathbf{U}_h = \text{Chol}(\mathbf{X}_{h-1}^*)$ for $h \geq 2$ where $\mathbf{X}_{h-1}^*$ is the optimal solution from the previous optimisation. $\mathbf{U} = \text{Chol}(\mathbf{X})$ represents the Cholesky decomposition of $\mathbf{X}$ such that $\mathbf{U}$ is an upper triangular matrix satisfying $\mathbf{X} = \mathbf{U}^T \mathbf{U}$.

Lemma 1. *Problem* (1.2) *gives an upper bound of Problem* (1.1).

Proof. The matrix $\mathbf{Q}$ is guaranteed to be PSD in (1.2), since it is symmetric and diagonally dominant with non-negative diagonal entries and therefore $\mathbf{X} = \mathbf{U}_h^T \mathbf{Q} \mathbf{U}_h$ is also PSD. Problem (1.2) minimises the objective function over a subset of PSD matrices and therefore Lemma 1 holds. □

Lemma 2. *Problem* (1.2) *is feasible for iteration h (h* ≥ 2*) if it is feasible for iteration* $h-1$.

Proof. This proof is trivial as $\mathbf{X}_{h-1}^* = \mathbf{U}_h^T \mathbf{I} \mathbf{U}_h$ and the identity matrix $\mathbf{I}$ is symmetric and diagonally dominant. Therefore, solution of problem in iteration $h-1$ is feasible for iteration h. □

Let us denote the objective function by $f(\cdot)$,

$$f(\mathbf{X}) = ||\mathbf{A} - \mathbf{X}||^2.$$

Lemma 1 and Lemma 2 indicate that an optimal solution always exists for (1.2) as long as the problem in the first problem is feasible and the optimal solution in each iteration is at least as good as that from the previous iteration, *i.e.,*

$$f(\mathbf{X}_h^*) \leq f(\mathbf{X}_{h-1}^*) \quad \forall h \geq 2.$$

Given that the problems are feasible and the optimal solution is PSD (which is true when numerical computation is concerned), it can be further shown that the optimal value of Problem (1.2) decreases strictly after each iteration unless it reaches the optimal value of Problem (1.1) [30]. In this case, the optimal objective value in each iteration in Problem (1.2) will finally converge as it is monotonically decreasing and bounded below by the true optimal value of Problem (1.1). In our numerical experiments, the optimal after each iteration always finally converges to the true optimal and strong empirical evidence shows a fast convergence.

Algorithm 1: An iterative quadratic/linear programming algorithm to solve PSDisation under some matrix norm

$\mathbf{X}_0 = \mathbf{I}$;

while *not converged* **do**

$\mathbf{U}_h = \text{Chol}(\mathbf{X}_{h-1})$;

find $\mathbf{X}_h$ by solving

$$\begin{aligned} \min_{\mathbf{X}_h, \mathbf{Q}_h} \ & ||\mathbf{A} - \mathbf{X}_h||^2 \\ s.t. \ & \text{diag}(\mathbf{X}_h) = \mathbf{1}, \\ & \mathbf{X}_h = \mathbf{U}_h^T \mathbf{Q}_h \mathbf{U}_h, \\ & \mathbf{Q}_h = \mathbf{Q}_h^T, \\ & \mathbf{Q}_h \text{ is diagonally dominant with non-negative diagonal;} \end{aligned}$$

$h = h + 1$;

end

Let us take the F-norm as an example. Note that $||\mathbf{A} - \mathbf{X}||_{\mathbf{F}}^2 = \boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{a}^T\boldsymbol{x} + \boldsymbol{a}^T\boldsymbol{a}$ where $\boldsymbol{x} = \text{vec}(\mathbf{X})$ and $\boldsymbol{a} = \text{vec}(\mathbf{A})$ and vec is the vectorisation function that transforms a matrix into a column vector. We further rewrite the objective function and the constraint that $\mathbf{Q}$ is diagonally dominant with non-negative diagonal by adding slack variables $\mathbf{R}$ into the optimisation,

$$\begin{aligned} \min_{\mathbf{X},\mathbf{Q},\mathbf{R}} \ & \frac{1}{2}\boldsymbol{x}^T\boldsymbol{x} - \boldsymbol{a}^T\boldsymbol{x} \\ s.t. \ & \text{diag}(\mathbf{X}) = \mathbf{1}, \\ & q_{ii} \geq \sum_{j \neq i} r_{ij}, && 1 \leq i \leq n \\ & -r_{ij} \leq q_{ij} \leq r_{ij}, && 1 \leq i \neq j \leq n \\ & \mathbf{X} = \mathbf{U}_h^T \mathbf{Q} \mathbf{U}_h, \\ & \mathbf{Q} = \mathbf{Q}^T, \end{aligned}$$

where q_{ij} and r_{ij} are the (i, j) entries of $\mathbf{Q}$ and $\mathbf{R}$, respectively. Now all constraints

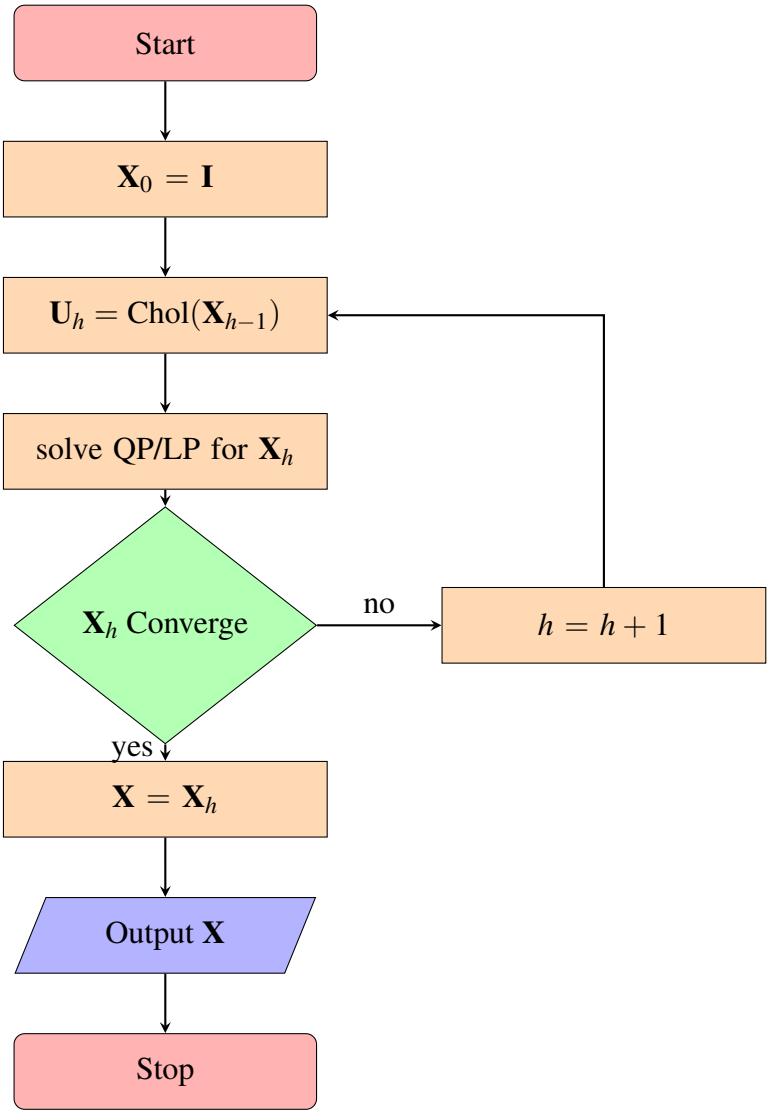

Figure 1.1: Flowchart of Algorithm 1: An iterative quadratic/linear programming algorithm to solve PSDisation under some matrix norm

become linear and we obtain a series of iterative QPs. Each QP has $3n^2$ variables and $4n^2 - n$ constraints, which can be solved efficiently.

Similarly, if the H-norm is applied in the objective function, the iterative QPs

can be written as

$$
\begin{aligned}
\min_{\mathbf{X},\mathbf{Q},\mathbf{R}} \ & \frac{1}{2}\boldsymbol{x}^T \operatorname{Diag}(\boldsymbol{h})\boldsymbol{x} - \boldsymbol{a}^T \operatorname{Diag}(\boldsymbol{h})\boldsymbol{x} \\
s.t. \ & \operatorname{diag}(\mathbf{X}) = \mathbf{1}, \\
& q_{ii} \geq \sum_{j \neq i} r_{ij}, && 1 \leq i \leq n \\
& -r_{ij} \leq q_{ij} \leq r_{ij}, && 1 \leq i \neq j \leq n \\
& \mathbf{X} = \mathbf{U}_h^T \mathbf{Q} \mathbf{U}_h, \\
& \mathbf{Q} = \mathbf{Q}^T.
\end{aligned}
$$

where $\boldsymbol{h} = \operatorname{vec}(\mathbf{H})$ and $\operatorname{Diag}(\boldsymbol{h})$ is an $n^2 \times n^2$ diagonal matrix with the elements of $\boldsymbol{h}$ on the main diagonal. This general formulation makes our method more flexible when an insurance company wants to assign weight to each entry of the correlation matrix, exploiting prior knowledge of the data.

Next, we consider the situation where we optimise with respect to the Chebyshev norm. According to [5], PSDisation over the F-norm is likely to result in a correlation matrix $\mathbf{X}$ in which some entries differ significantly from the initial matrix $\mathbf{A}$, while other entries have relatively smaller deviations, giving the minimum F-norm optimal. However, this is not preferred by some insurance companies as they aim to minimise the maximum discrepancy between corresponding entries from the valid correlation matrix $\mathbf{X}$ and the initial matrix $\mathbf{A}$. In this case, minimising over the Chebyshev norm becomes an alternative choice. With our iterative approximating

method, this can be obtained by solving a series of LPs of the following form,

$$
\begin{aligned}
&\min_{\mathbf{X},\mathbf{Q},\mathbf{R},t} \; t \\
&\quad s.t. \; -t \leq a_{ij} - x_{ij} \leq t, && 1 \leq i \neq j \leq n \\
&\qquad \text{diag}(\mathbf{X}) = \mathbf{1}, \\
&\qquad q_{ii} \geq \sum_{j \neq i} r_{ij}, && 1 \leq i \leq n \\
&\qquad -r_{ij} \leq q_{ij} \leq r_{ij}, && 1 \leq i \neq j \leq n \\
&\qquad \mathbf{X} = \mathbf{U}_h^T \mathbf{Q} \mathbf{U}_h, \\
&\qquad \mathbf{Q} = \mathbf{Q}^T.
\end{aligned}
$$

Note that in this formulation, minimising $||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}^2$ can be equivalently done by minimising $||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}$ and thus only LPs instead of QPs need to be solved.

Experiments show that our iterative algorithm proposed for approximating optimal solutions for PSDisation problems offers great flexibility under different norms while maintaining a good accuracy. Details are explained in Section 1.5.

1.4.2 Gradient Descent

Next we propose our second algorithm, where we will focus on working under the F-norm only as there is currently no known method that could project $\mathbf{X}$ onto the PSD set under the H-norm. This method is therefore less flexible in the choice of norms. However, it works more efficiently and provides more steady results on real data with noise.

Consider the objective in (1.1), and note that the gradient of its objective function under the F-norm is $\frac{\partial}{\partial \mathbf{X}}||\mathbf{A}-\mathbf{X}||_{\mathbf{F}} = \frac{\mathbf{X}-\mathbf{A}}{||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}}$. To project a matrix $\mathbf{X}$ onto the PSD cone $\mathscr{S}_+^n$, first perform the eigendecomposition such that $\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$ where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is the square matrix whose columns are eigenvectors of $\mathbf{X}$, and $\mathbf{\Lambda} = \text{Diag}(\boldsymbol{\lambda})$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues λ_i of $\mathbf{X}$. Define $\mathbf{\Lambda}_+ = \text{Diag}(\boldsymbol{\lambda}_+)$ where $\lambda_{+i} = \max(\lambda_i, 0)$. Next, in order to ensure that the resulting matrix has diagonal $\mathbf{1}$, we calculate the scaling matrix

$\mathbf{T} = \text{Diag}(\boldsymbol{t})$ where the weighting parameter t_i is given by

$$t_i = \left(\sum_{m=1}^{n} q_{im}^2 \lambda_{+m} \right)^{-1/2}.$$

Then $\mathbf{X}_+ = \mathbf{T}\mathbf{Q}\mathbf{\Lambda}_+\mathbf{Q}^{-1}\mathbf{T}$ is the projection of $\mathbf{X}$ onto the cone of PSD matrices.

Our proposed method is shown in Algorithm 2 below. $\mathbf{X}$ is updated iteratively by taking a gradient step followed by a projection onto the PSD cone until converged. According to our experiments, the gradient step γ can be initialised as a large value to speedup the convergence and it will fast decrease during iterations.

Algorithm 2: The Gradient descent algorithm for PSDisation under the F-norm

$\mathbf{X}_0 = \mathbf{I}$;
while *not converged* **do**
  $\mathbf{X}_{i+1} = \mathbf{X}_i - \gamma \mathbf{G}_i$;
  $\mathbf{X}_{i+1} = \mathscr{P}_{\mathscr{S}^+}(\mathbf{X}_{i+1})$;
  $C_{i+1} = ||\mathbf{A} - \mathbf{X}_{i+1}||_{\mathbf{F}}$;
  if $C_{i+1} < C_i$ **then**
    $\gamma = \gamma * (1 + \delta)$;
  else
    $\gamma = \gamma / 2$;
  end
  $i = i + 1$;
end

1.5 Experimental Results on Simulated Data

To illustrate the robustness and flexibility of our algorithms in different settings, we designed a series of tests based on both simulated and real datasets. In this section, we explain and show the experimental results on simulated data and compare our methods with the state-of-the-art methods. In the next section, we give examples of some practical applications. All experiments were run using MATLAB R2023a Prerelease (9.14.0.2137306) with Financial Toolbox v6.5, Optimization Toolbox

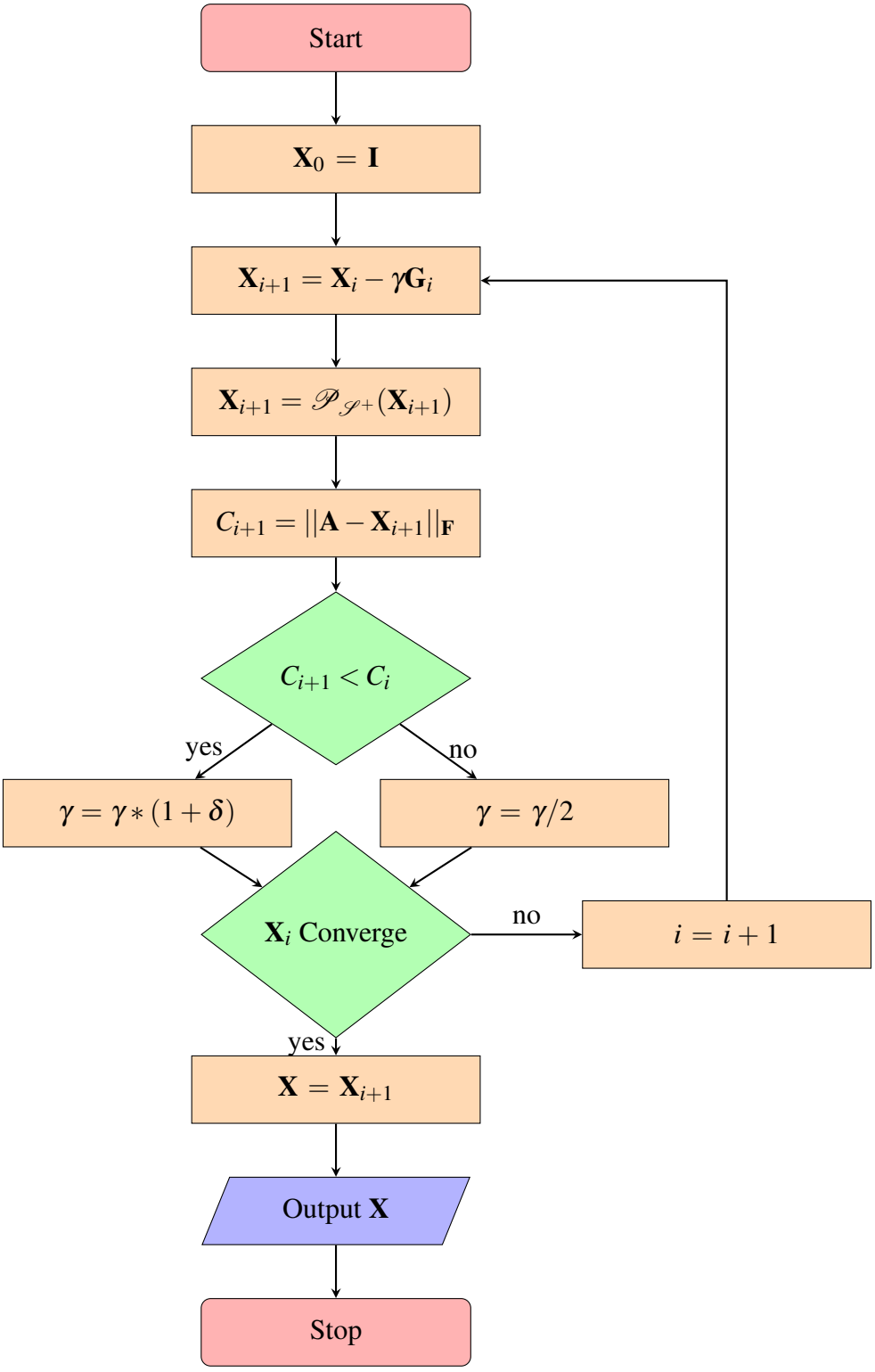

Figure 1.2: Flowchart of Algorithm 2: The Gradient descent algorithm for PSDisation under the F-norm

v9.5, and Statistics and Machine Learning Toolbox v12.5, and Gurobi Optimizer 10.0.1.

1.5.1 Experiments on Iterative Quadratic/Linear Programming Algorithm with Simulated Data

To test our iterative quadratic/linear programming algorithm, we randomly generate the initial matrix $\mathbf{A}$ of different dimensions (*i.e.* with dimensions of 5, 10, 25, 50 and 75) that is not PSD. For $\mathbf{A}$ generated from each dimension settings, we solve for the nearest valid correlation matrix $\mathbf{X}$ using our method under the F-norm and the Chebyshev norm, which lead to iterative quadratic programming (IQP) and iterative linear programming (ILP), respectively. The stopping criteria is set to achieve a solution with error less than 0.1% in each iteration. We calculate the Frobenius distance and the Chebyshev distance between $\mathbf{X}$ and $\mathbf{A}$ from the IQP and ILP and compare with those given by the APM, the Newton method and the SDM, which are all designed for minimising the F-norm. The APM and the Newton method work very fast and can both achieve solutions with accuracy tolerance far lower than 0.0001, therefore the experiment settings have no big impact on testing results. We just use the default settings by the authors here [6, 8]. The results are shown in Tables 1.1 and 1.2 for the F-norm and the Chebyshev norm, respectively. The best performances in each group are shown in bold.

Table 1.1: The F-norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}$ obtained from different algorithms on different dimensions of the initial matrix $\mathbf{A}$. (Best performances are marked in bold.)

Dimension of A	**5**	**10**	**25**	**50**	**75**
Newton	**0.3420**	**0.8341**	**5.0057**	**13.0331**	**20.6889**
APM	**0.3420**	**0.8341**	**5.0057**	**13.0331**	**20.6889**
SDM	0.3537	0.8602	5.1716	13.5887	21.5025
IQP	**0.3420**	0.8352	5.0245	13.2406	21.2224
ILP	0.3819	1.0815	7.6554	15.9193	24.9860

Table 1.2: The Chebyshev norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}$ obtained from different algorithms on different dimensions of the initial matrix $\mathbf{A}$. (Best performances are marked in bold.)

Dimension of A	**5**	**10**	**25**	**50**	**75**
Newton	0.1313	0.2128	0.6473	0.7605	0.8313
APM	0.1313	0.2128	0.6473	0.7605	0.8313
SDM	0.1416	0.2265	0.6225	0.7278	0.7512
IQP	0.1304	0.2131	0.6245	0.7314	0.8397
ILP	**0.0854**	**0.1375**	**0.4293**	**0.4310**	**0.4805**

To give a clearer view on how these algorithms compare with each other, we use the Frobenius distance and the Chebyshev distance obtained from the Newton method as a benchmark and calculate the relative change of the two distances obtained from other algorithms in percentage. For example, the relative change of the Frobenius distance obtained from the IQP algorithm compared to the Newton method is given by $\frac{||\mathbf{A}-\mathbf{X}_{\text{IQP}}||_{\mathbf{F}}-||\mathbf{A}-\mathbf{X}_{\text{Newton}}||_{\mathbf{F}}}{||\mathbf{A}-\mathbf{X}_{\text{Newton}}||_{\mathbf{F}}} \times 100\%$. In Tables 1.3 and 1.4, a negative percentage means that the optimal solution obtained from this method is better than that from the Newton method in the corresponding norm while a positive percentage represents a worse result.

Table 1.3: Relative change in the F-norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}$ compared with the Newton method in percentage. (Best performances are marked in bold.)

Dimension of A	**5**	**10**	**25**	**50**	**75**
Newton	**0**	**0**	**0**	**0**	**0**
APM	**0**	**0**	**0**	**0**	**0**
SDM	3.43	3.13	3.31	4.26	3.93
IQP	0.02	0.14	0.3763	1.59	2.58
ILP	11.67	29.66	52.93	22.15	20.77

Table 1.4: Relative change in the Chebyshev norm $||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}$ compared with the Newton method in percentage. (Best performances are marked in bold.)

Dimension of A	**5**	**10**	**25**	**50**	**75**
Newton	0	0	0	0	0
APM	0	0	0	0	0
SDM	7.79	6.45	-3.84	-4.30	-9.64
IQP	-0.70	0.16	-3.52	-3.82	1.01
ILP	**-34.98**	**-35.37**	**-33.68**	**-43.33**	**-42.21**

Furthermore, we report the computing time of each run in Table 1.5. The lowest running times are shown in bold.

Table 1.5: Running time of different algorithms on different dimensions of the initial matrix **A** (in seconds). (Best performances are marked in bold.)

Dimension of A	**5**	**10**	**25**	**50**	**75**
Newton	0.0024	0.0038	0.0058	0.0062	0.0350
APM	0.0005	0.0010	0.0068	0.0281	0.0615
SDM	**0.0001**	**0.0001**	**0.0003**	**0.0008**	**0.0025**
IQP	0.0315	0.1260	3.84	546	8180
ILP	0.0206	0.0937	2.35	719	5661

It can be easily concluded from Table 1.3 (and also see Table 1.5 for efficiency) that both the APM and the Newton method give solutions with the minimum Frobenius distance, which they aim to minimise, while the Newton method works slightly faster in high dimensions. (Further experiments show that the ALD approach also gives the same optimal solutions, which we omit in the above tables.) The SDM on the other hand is not as accurate as the Newton method and the APM but is the fastest algorithm overall as it does not require iterations. The IQP method gives solutions slightly worse than the Newton method in terms of the F-norm but the difference is not significant, especially in low dimensions. These results from the IQP can be improved by setting a lower tolerance and hence increasing the number of iterations but will cost more time to converge. The ILP method is designed to minimise with respect to the Chebyshev norm and therefore does not give solid results in the F-norm as other algorithms mentioned above. However, the nearest correlation matrix generated by the ILP method has a much lower Chebyshev distance than by the Newton method which intends to minimise the F-norm (see Table 1.4). This result indicates that minimising over the Chebyshev norm can indeed be a good alternative for PSDisation since considering the F-norm will potentially result in huge deviations in some entries of **X**, which are not favoured by some insurance companies [5]. It is worth mentioning that both the IQP and ILP methods take a long time to run when the dimension of **A** is high, this could potentially be improved by optimising the QP and LP solvers in the future work. For example, possible

improvements could be to carefully track the optimisations and keep an active set to significantly reduce the number of constraints, or to manually set a starting point for each optimisation based on the solution from the previous iteration.

To illustrate the flexibility of our IQP algorithm and its potential to achieve good accuracy, our next step is to test when weights are assigned to each coefficient, *i.e.* to work under the H-norm. Similarly as before, we generate the initial matrix $\mathbf{A}$ of different dimensions. This time, we apply the ALD method and our IQP algorithm to find the nearest correlation matrix to $\mathbf{A}$ under the F-norm and the H-norm, respectively, where each entry of the weight matrix $\mathbf{H}$ is randomly generated from a uniform distribution. We also test these methods on PSDisation problems with constraints, that is, we let $x_{ij} = 0$ if $|a_{ij}| < 0.1$, we also request $x_{ij} > 0$ if $a_{ij} > 0.5$ and $x_{ij} < 0$ if $a_{ij} < -0.5$. Table 1.6 shows the optimal values under different settings.

Table 1.6: Optimal solution by ALD and IQP for constrained and unconstrained PSDisation problems under the F-norm and the H-norm on different dimensions of the initial matrix $\mathbf{A}$. (Best performances are marked in bold.)

Dimension n				10	20	40	70
# of equality constraints (=0)				4	19	74	263
# of inequality constraints (<0/>0)				25	92	367	1208
Optimal solution	Unconstrained	F-norm	ALD	**2.5032**	**6.2625**	**14.8958**	**30.2514**
			IQP	2.5060	6.2789	15.1565	30.7230
		H-norm	ALD	**1.6149**	**3.9136**	**10.0493**	**20.3044**
			IQP	1.6188	3.9360	10.2596	20.8868
	Constrained	F-norm	ALD	**2.5401**	**6.4399**	**15.2334**	**30.8318**
			IQP	2.5411	6.4562	15.3771	31.2218
		H-norm	ALD	1.6631	4.0789	10.4666	**20.9562**
			IQP	**1.6501**	**4.0707**	**10.4355**	21.1677

It can be concluded that both ALD and IQP work effectively on constrained or unconstrained PSDisation problems based on the F-norm or the H-norm. Our IQP algorithm runs slowly when dimension is high and produces slightly worse results than ALD, but the optimal values are still competitive. For example, when working under the F-norm with dimension $n = 10$, IQP gives optimal solutions of 2.5060 and 2.5411 in the unconstrained and constrained PSDisation tasks, respectively, which are close to those given by ALD, 2.5032 and 2.5401. On the other hand, IQP gives

slightly better results than ALD in constrained problems with the H-norm when the dimension is low. For example, in the constrained task under the H-norm with $n = 10$, the optimal solution given by IQP is 1.6501, which is better than 1.6631 by ALD. In addition, our iterative algorithm shows more flexibility when problem settings are combined with the H-norm (IQP) and the Chebyshev norm (ILQ), where the ALD method becomes infeasible.

1.5.2 Experiments on Gradient Descent Method with Simulated Data

In order to test the performance of the gradient descent method on PSDisation problems, we compare ALD, IQP, GD and SDM on unconstrained problems of different dimensions under the F-norm. SDM is a trivial method that project the matrix onto the PSD cone directly and therefore can be considered as a baseline for this experiment. Table 1.7 compares the convergence times and optimal solutions of all four algorithms.

Table 1.7: Convergence time and optimal solution by ALD, IQP, GD and SDM for unconstrained PSDisation problems under the F-norm on different dimensions of the initial matrix **A**. (Best performances are marked in bold.)

Dimension n		10	20	40	70	100	200
Convergence time (unconstrained, F-norm)	ALD	0.0029s	0.0027s	0.0120s	0.0076s	0.0163s	0.0364s
	IQP	0.17s	1.52s	2min 19s	2h 29min	-	-
	GD	0.03s	0.06s	0.16s	0.07s	0.08s	0.26s
	SDM	**0.0003s**	**0.0004s**	**0.0007s**	**0.0011s**	**0.0017s**	**0.0047s**
Optimal solution (unconstrained, F-norm)	ALD	**2.5032**	**6.2625**	**14.8958**	**30.2514**	**44.9199**	**96.3681**
	IQP	2.5060	6.2789	15.1565	30.7230	-	-
	GD	2.5316	6.2754	15.0992	30.8536	46.0570	98.7712
	SDM	2.5667	6.5599	15.5756	31.3282	46.5758	99.3325

As shown in Table 1.7, GD provides comparable results to ALD. Though slightly slower than ALD, GD works much more efficient than general SDP solvers. Furthermore, we want to illustrate the advantages of the gradient descent (GD) method over the ALD algorithm. Consider the situation where in practice, the correlation matrices obtained by some insurance companies are usually inaccurate. We expect our PSDisation algorithm to work better when error exists in the input matrix according to the testing results.

In the following experiment, we randomly generate a correlation matrix $\mathbf{A}$ that is not PSD. Noise from normal distribution is then added to each non-diagonal entry of $\mathbf{A}$ and $\mathbf{A}'$ represents the estimated correlation matrix. We use GD, ALD and IQP to calculate the nearest PSD correlation matrix $\mathbf{X}$ to $\mathbf{A}'$ under the F-norm and compare $||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}$, the distance between $\mathbf{X}$ and the actual initial matrix $\mathbf{A}$.

Table 1.8: Average Frobenius distance from $\mathbf{X}$ to $\mathbf{A}$ and $\mathbf{A}'$ by different algorithms over 100 tests in experiment settings. (Best performances are marked in bold.)

	$\|\|\mathbf{A}'-\mathbf{X}\|\|_{\mathbf{F}}$	$\|\|\mathbf{A}-\mathbf{X}\|\|_{\mathbf{F}}$
ALD	**4.43**	5.70
IQP	4.45	5.72
GD	4.60	**5.49**

Table 1.8 shows the average results of the three algorithms tested on 100 different initial matrices of dimension 40 with noise of standard deviation 0.2. It can be concluded that while GD provides slightly inaccurate solutions to the PSDisation optimisation problem, it is less influenced by noise, and thus leads to better results than ALD overall. In all 100 testing examples, GD gives matrix $\mathbf{X}$ that is closer to the initial matrix $\mathbf{A}$. In practice, GD can be more resilient to noise with a large standard deviation than ALD. Details of the experimental results are given in Figure 1.3.

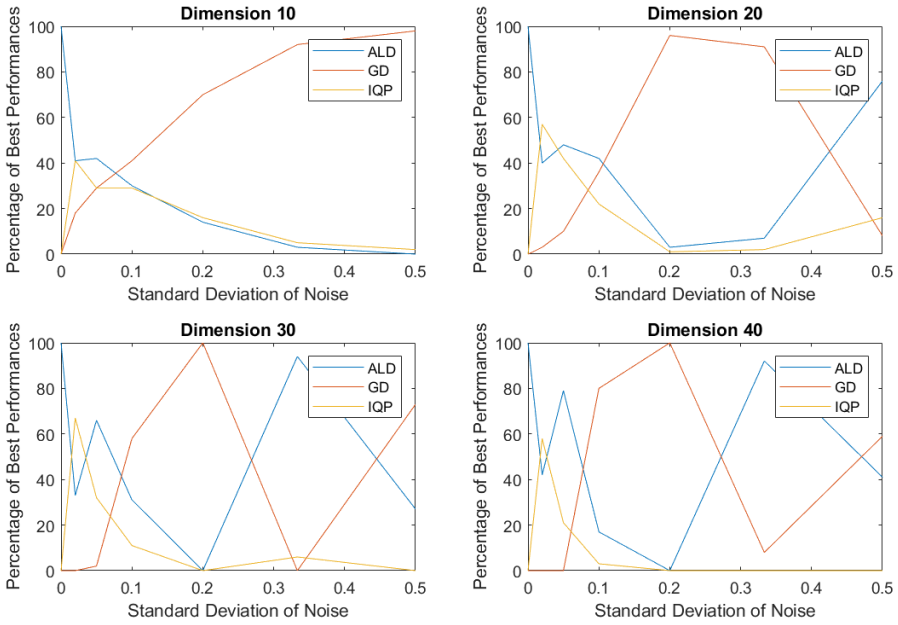

Figure 1.3: Percentage of best performances by ALD, GD and IQP versus standard deviation of the noise in PSDisation tests with different dimensions of the initial matrix $\mathbf{A}$.

Figure 1.3 plots the percentages of the best performance each PSDisation algorithm achieves with different dimensions of the correlation matrix and different standard deviations of noise. The best performance is decided by giving the shortest distance between $\mathbf{X}$ and $\mathbf{A}$, where $\mathbf{X}$ is calculated by applying different PSDisation algorithms on the noised matrix $\mathbf{A}'$. The tests are repeated 100 times for matrices of different dimensions and noise of different levels.

It can be concluded that despite the fact that ALD performs best on accurate initial matrices (when noise does not exist), IQP and GD can be good alternatives when the input comes with noise, especially for matrices with large dimension. IQP works well when the noise is small, *e.g.* in 67% of our tests IQP gives the smallest $||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}$ for matrices with dimension 30 and noise with standard deviation 0.02. When the noise becomes slightly larger (for a standard deviation between 0.1 and 0.2), GD performs significantly better.

1.5.3 Experiments with Simulated Data Rounded to Multiples of 25%

We present results of comparisons between ALD, IQP, GD and SDM on PSDisation problems under the F-norm in practical settings where entries of the correlation matrix are rounded. Without knowing the actual correlations, the way the European Commission creates the correlation matrix for the Solvency Capital Requirement insurance models is by picking the best possible choices for the correlation matrix entries from the set $\{-75\%, -50\%, -25\%, 0\%, 25\%, 50\%, 75\%\}$. The correlation matrix $\mathbf{A}$ estimated this way is usually not PSD, but certain entries of $\mathbf{A}$ can be expected to be positive or negative as actuaries expect some risks to be positive or negative correlated based on domain knowledge.

Inspired by the above practical scenario, our experiment is designed as follows. Firstly a PSD matrix $\mathbf{A}_t$ is generated which we assume is the true correlation matrix. Then 250 observations are generated from t-distribution with 3 degrees of freedom and are used to calculate the sample correlation matrix $\mathbf{A}_s$. We obtain our estimated correlation matrix $\mathbf{A}'_s$ by rounding each entry of the sample correlation matrix $\mathbf{A}_s$ to 25%. We also round the true correlation matrix $\mathbf{A}_t$ to 25% to get $\mathbf{A}'_t$. Now $\mathbf{A}'_s$ and $\mathbf{A}'_t$ are not PSD. We perform an unconstrained PSDisation on $\mathbf{A}'_s$ under the F-norm and the solution is denoted as $\mathbf{X}$. We also test constrained PSDisation by ALD and IQP. The constraints are set so that the entries of $\mathbf{X}$ have the same signs as $\mathbf{A}_s$, which we assume are available to actuaries as domain knowledge.

The tests are done repeatedly on matrices of dimensions 10, 20, 30 and 40. The unconstrained PSDisations are done using ALD, IQP, GD and SDM. In our experiment settings, only around 85% to 89% of the non-diagonal entries of $\mathbf{X}$ have the same sign as $\mathbf{A}_s$. In the constrined settings, both ALD(con) and QP(con) can 100% satisfy the same sign constraints, either strictly or to some small tolerance. The results are shown in Tables 1.9 and 1.10.

Table 1.9: Average Frobenius distance from $\mathbf{X}$ to $\mathbf{A}'_s$ and $\mathbf{A}'_s$ by different algorithms over 500 tests in experiment settings. (Best performances are marked in bold.)

	n	**10**	**20**	**30**	**40**
$\|\|\mathbf{A}'_s - \mathbf{X}\|\|_\mathbf{F}$	**ALD**	**0.1387**	**0.3940**	**0.6991**	**1.0320**
	IQP	0.1388	0.3945	0.7003	1.0347
	GD	0.1453	0.4198	0.7512	1.1252
	SDM	0.1462	0.4247	0.7633	1.1479
	ALD(con)	0.1421	0.4081	0.7262	1.0735
	IQP(con)	0.1421	0.4086	0.7276	1.0760
$\|\|\mathbf{A}_s - \mathbf{X}\|\|_\mathbf{F}$	**ALD**	0.6429	1.2817	1.8862	2.4612
	IQP	0.6429	1.2820	1.8870	2.4633
	GD	0.6433	1.2837	1.8910	2.4796
	SDM	0.6440	1.2887	1.9019	2.5017
	ALD(con)	0.6351	**1.2499**	**1.8217**	**2.3611**
	IQP(con)	**0.6350**	1.2501	1.8223	2.3624

Table 1.10: Average Frobenius distance from $\mathbf{X}$ to $\mathbf{A}'_t$ and $\mathbf{A}'_t$ by different algorithms over 500 tests in experiment settings. (Best performances are marked in bold.)

	n	**10**	**20**	**30**	**40**
$\|\|\mathbf{A}'_t - \mathbf{X}\|\|_\mathbf{F}$	**ALD**	1.6960	3.5396	5.2255	7.0881
	IQP	1.6960	3.5398	5.2261	7.0896
	GD	1.6851	3.4839	5.1133	6.8849
	SDM	**1.6814**	**3.4640**	**5.0635**	**6.7970**
	ALD(con)	1.6923	3.5240	5.1953	7.0412
	IQP(con)	1.6923	3.5240	5.1959	7.0425
$\|\|\mathbf{A}_t - \mathbf{X}\|\|_\mathbf{F}$	**ALD**	1.5294	3.2316	4.7367	6.4601
	IQP	1.5294	3.2318	4.7373	6.4617
	GD	1.5169	3.1710	4.6136	6.2364
	SDM	**1.5128**	**3.1499**	**4.5599**	**6.1427**
	ALD(con)	1.5252	3.2135	4.7015	6.4052
	IQP(con)	1.5252	3.2138	4.7022	6.4067

It can be easily concluded from Table 1.9 that ALD is the best in minimising the objective, $||\mathbf{A}'_s - \mathbf{X}||_\mathbf{F}$, while IQP could produce very similar results. On the other hand, GD gives slightly worse solutions than ALD and IQP, but still better than baseline method SDM. Comparing $||\mathbf{A}_s - \mathbf{X}||_\mathbf{F}$, unsurprisingly, we see that constrained PSDisation shows its advantages if prior information is available. While ALD allows restrictions on upper or lower bounds of entries of $\mathbf{X}$, IQP could potentially allow for any linear constraints and thus can be more flexible. Nonlinear

constraints can also be set if efficiency is not the main focus. For example, we could restrict the entries of $\mathbf{X}$ to be multiples of 25%. This leads to iterative mixed-integer programming, which is beyond the scope of this thesis.

On the other hand, comparing $||\mathbf{A}'_t - \mathbf{X}||_{\mathbf{F}}$ and $||\mathbf{A}_t - \mathbf{X}||_{\mathbf{F}}$ in Table 1.10, SDM and GD constantly give PSD matrix $\mathbf{X}$ that are closer to the true correlation matrix $\mathbf{A}_t$. Therefore, we draw the conclusion that although relatively inaccurate in minimising the objective function, $||\mathbf{A}'_s - \mathbf{X}||_{\mathbf{F}}$, in practice, SDM and GD are more resilient to noise generated in real data, which coincides with our observations in the previous part.

1.6 Experiments Results on Real Datasets

The potential of our two proposed PSDisation methods has been illustrated with synthetic/simulated data. In this section, we test them with real-life data with applications in (financial) portfolio construction and dimension (data) reduction. We compare our IQP/ILP and GD with other state-of-the-art PSDisation algorithms. The shrinkage method is also included as it is widely used in finance, econometrics, and statistics to tackle the ill-conditioned or non-PSD matrix problem.

1.6.1 Experiments with Assets Daily Return Data

We now present an example of PSDisation applications with real-life financial data that consist of 50 public companies from NASDAQ stock market that started to be listed on this marked at different points in time, i.e., assets have different observation periods that depend on first trading day of each asset. That is, we use the daily asset returns since 2010 to calculate the pairwise correlation. This is useful when building a portfolio and calculating the portfolio's (aggregate) risk position measured through risk measures such as Value at Risk (VaR) and Conditional Value at Risk (CVaR). However, the resulting correlation matrix $\mathbf{A}$ is non-PSD and has negative eigenvalues since i) missing values exist for each stock at certain periods of time, and ii) each asset pair has different overlapping observation period. A solution to this issue is to use PSDisation algorithms as portfolio construction means computing weights for each asset and such computation requires PSD empirical

correlation matrix. Alternatively, we can use some imputation techniques to replace the missing data with substituted values, but this does not guarantee a PSD empirical covariance matrix due to ii). Imputation techniques are statistically based that require no domain knowledge or one may rely on imputation methodologies that are specific to financial applications; *e.g.* one may replace the missing returns with risk-free returns which is the rate of the 10-year rated government Treasury note. Such imputations are the last resort solution with an unknowable impact on the application in hand.

We perform PSDisation on the calculated correlation matrix $\mathbf{A}$ to get $\mathbf{X}$. The testing algorithms include IQP, ILP, GD, ALD, and SDM. The shrinkage method [24] is not designed to work with missing data. Therefore we impute the data before applying the shrinkage method. We use two methods to fill the missing entries, namely replacing them with mean observed values of each variable and filling using the multivariate imputation by chained equations method (MICE). Replacing with mean value is a simple approach to handle missing data, but may not be appropriate for estimating correlations when there are too many missing entries. MICE uses a sequence of conditional models for imputation so that each variable is modelled conditionally on other variables. The shrinkage method is then used to estimate the correlation matrix $\mathbf{X}$ and we call them Shrinkage-Mean and Shrinkage-MICE for the two different imputation strategies. We calculate the Frobenius distance and the Chebyshev distance between $\mathbf{A}$ and $\mathbf{X}$ and the results are shown in Table 1.11.

Table 1.11: Frobenius distance and Chebyshev distance between $\mathbf{A}$ and $\mathbf{X}$ by different algorithms on NASDAQ stocks return correlation matrix. (Best performances are marked in bold.)

	$\|\|\mathbf{A}-\mathbf{X}\|\|_{\mathbf{F}}$	$\|\|\mathbf{A}-\mathbf{X}\|\|_{\mathbf{C}}$
ILP	1.0836	**0.0236**
IQP	0.3765	0.0872
ALD	**0.3764**	0.0886
GD	0.3958	0.0836
SDM	0.4085	0.0931
Shrinkage-Mean	2.5625	0.4025
Shrinkage-MICE	2.3884	0.4367

The same experiment was also carried out with mixed choices of top 25 SP500 companies and 25 NASDAQ stocks where the correlation matrix calculated is invalid. The results are shown in Table 1.12.

Table 1.12: Frobenius distance and Chebyshev distance between **A** and **X** by different algorithms on a mixture of NASDAQ and SP500 stocks return correlation matrix. (Best performances are marked in bold.)

	$\|\|\mathbf{A}-\mathbf{X}\|\|_{\mathbf{F}}$	$\|\|\mathbf{A}-\mathbf{X}\|\|_{\mathbf{C}}$
ILP	0.5605	**0.0121**
IQP	0.2269	0.0542
ALD	**0.2268**	0.0547
GD	0.2488	0.0602
SDM	0.2534	0.0620
Shrinkage-Mean	2.4080	0.3935
Shrinkage-MICE	1.8739	0.3142

It can be concluded from Tables 1.11 and 1.12 that the optimal minimum Frobenius distance solution by our IQP method is very close to the state-of-the-art ALD method, while maintaining an even lower Chebyshev distance. This ensures that the maximum deviation of correlation for each pair of stocks by IQP is smaller than that by ALD. We also noticed that in this real financial data experiment, the speed of convergence of IQP is faster and the optimal solution is closer to ALD compared with those in previous simulated data, providing evidence that IQP is potentially suitable for dealing with real dataset. In addition, GD performs slightly better than the baseline method SDM in this stocks data experiment in terms of both the F-norm and the Chebyshev norm. In Table 1.11 GD achieves even better Chebyshev norm with NASDAQ stocks than ALD, IQP and SDM, although this is not guaranteed in all experimental settings. ILP gives a much lower Chebyshev norm than other methods as expected, since it is designed to do so. The shrinkage method is not designed to minimise any of the norms and thus does give comparable results under these criteria. However, it aims to give a more stable and well-conditioned estimate of the correlation matrix. We now evaluate how the above methods perform in the application of portfolio optimisation.

We divide the above assets return data into two parts. Data since January 2010

until June 2021 serves as historical observations to estimate the correlation matrix using either PSDisation algorithms or the shrinkage method after imputation. We then build a minimum variance portfolio with the correlation matrix from each method and test how it performs in the following 20 months. The annualised average return, annualised standard deviation, and the Sharpe ratio for the NASDAQ stock portfolio and the mixed portfolio are shown in Tables 1.13 and 1.14. An equal weight portfolio is provided as a benchmark.

Table 1.13: Annualised average return, annualised standard deviation and Sharpe ratio of NASDAQ stocks minimum variance portfolios based on correlation matrices estimated using different methods. (Best performances are marked in bold.)

Method	Average Return (%)	Standard Deviation (%)	Sharpe Ratio
ALD	-15.6085	19.7047	-0.7921
GD	-14.7107	19.3380	-0.7607
ILP	-13.6843	20.6439	-0.6629
IQP	-15.6040	19.7084	-0.7917
SDM	-14.6754	19.3006	-0.7604
Shrinkage-Mean	**-11.5543**	**18.0076**	-0.6416
Shrinkage-MICE	-12.9444	18.7709	-0.6896
Equal weight	-18.9210	31.2105	-0.6062

Table 1.14: Annualised average return, annualised standard deviation and Sharpe ratio of a mixture of NASDAQ and SP500 stocks minimum variance portfolios based on correlation matrices estimated using different methods. (Best performances are marked in bold.)

Method	Average Return (%)	Standard Deviation (%)	Sharpe Ratio
ALD	6.5146	15.1593	0.4297
GD	6.5242	15.1410	0.4309
ILP	**7.0063**	15.2558	**0.4593**
IQP	6.5072	15.1598	0.4292
SDM	6.5160	**15.1395**	0.4304
Shrinkage-Mean	3.4394	15.1660	0.2268
Shrinkage-MICE	6.1097	15.3055	0.3992
Equal weight	-2.6196	24.7886	-0.1057

Judging from annualised standard deviation, we see that all minimum variance portfolios have much lower volatility compared to the equal weight portfolio. In the NASDAQ portfolio (Table 1.13), the portfolios built with the shrinkage method achieve both higher annualised return and lower standard deviation. Portfolios with

GD and SDM also have comparable results. However, since the annualised return is negative, it is not easy to determine which is the best. In the mixed portfolio (Table 1.14), SD achieves the lowest standard deviation, while the portfolio with ILP gives the best return and Sharpe ratio. We would like to point out that the shrinkage method only works without missing values in the data. Therefore, different data imputation methods can influence its performance. Table 1.14 gives a good example of this where Shrinkage-Mean and Shrinkage-MICE portfolios show a significant difference in their returns. Another note to make is that in this experiment, all portfolios are built with correlation matrices estimated using historical data and are then fixed for 20 months. The results might be different if the correlation matrices are estimated based on moving windows and the portfolios are updated over time.

1.6.2 Experiments with Machine Learning Data for Applications in PCA

Principal component analysis (PCA) [3] is a popular statistical tool to reduce the dimension of the dataset while maintaining as much information as possible from the data, which helps visualising the data or performing further actions in different fields of studies. The covariance matrix of the features is calculated before it turns into an eigenvalue problem [31]. A standardisation of the data is usually needed when the features are in different scales or, alternatively, the correlation matrix can be used instead of the covariance matrix. If missing values present in the dataset, a standard way is to delete all instances that contain missing values and then calculate the correlation matrix. However, ignoring such instances means that less information can be used from the whole dataset, which may lead to a decrease of the quality of PCA. We seek to use as much information as possible from the dataset by calculating the correlation between attributes pairwisely. This may cause the resulting correlation matrix to be non-PSD and therefore a PSDisation process is needed.

We use the *wine* data available from the UCI Machine Learning Repository [32] as an example to show the advantages of using PSDisation on pairwisely calculated correlation matrix. The data consists of 13 attributes of chemical analysis of wines from different cultivars in Italy. The target is categorical with three classes.

The features are in different scales so it makes sense to use the correlation matrix in PCA. We first drop 20% of the values from the dataset randomly so that it has missing values. We calculate the pairwised correlation matrix which is non-PSD and thus invalid. Then PSDisation is performed on the invalid correlation matrix, after which eigenvectors are calculated to transfer the standardised data with missing values. The variance of each principal component of the transformed data is reported in Table 1.15. We use IQP algorithm for PSDisation in this example. We also perform PCA according to the correlation matrix calculated by omitting all instances that have missing values. Figure 1.4 shows the cumulative sum of the variance for each principal component of the transformed data by different methods, where *all PCA* stands for PCA performed with all instances with missing values omitted and *pairwise PCA* stands for PCA using pairwised correlation matrix with PSDisation. Since PCA aims to capture as large variance in the data via the top several principal components as possible, a curve that increases faster within the first several principal components indicates a better PCA result.

Table 1.15: Variance of each principal component in the transformed wine data. (Higher variance in each row is marked in bold.)

Principal Component	all PCA	pairwise PCA
1	3.1258	**3.2692**
2	1.5018	**1.6485**
3	0.7359	**1.0723**
4	0.6901	**0.7908**
5	**0.7454**	0.7164
6	0.5511	**0.5894**
7	**0.7294**	0.5543
8	**0.4103**	0.3897
9	**0.4756**	0.3526
10	**0.3532**	0.2875
11	**0.3712**	0.2895
12	**0.3508**	0.2604
13	**0.3436**	0.1637

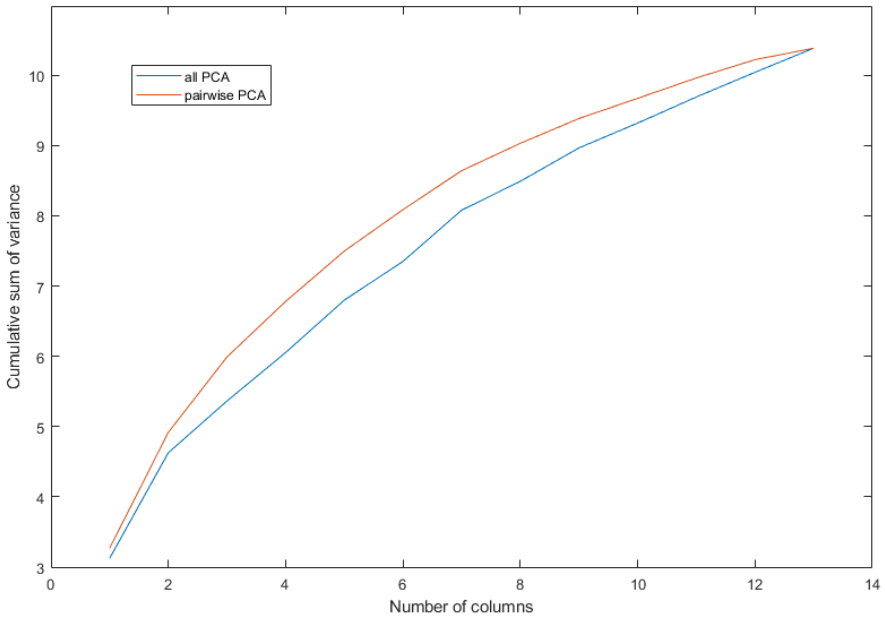

Figure 1.4: Cumulative sum of the variance of each principal component in the transformed wine data.

Table 1.15 indicates that with the use of pairwised correlation, the transformed data has higher variance in the first several principle components than all PCA. Therefore, more important information can be captured by the first few principal components via pairwise PCA.

Table 1.16: Weight of each feature assigned by the 1_{st} and 2_{nd} principal components in the wine data by PCA, pairwise PCA and all PCA. Top: Weights of features in first principal component. Bottom: Weights of features in second principal component. (Top 3 weights in each column are marked in bold.)

Features	PCA	pairwise PCA	all PCA
Flavanoids	**0.4229**	**0.4212**	**0.3793**
Total phenols	**0.3947**	**0.3753**	**0.3494**
OD280/OD315 of diluted wines	**0.3762**	**0.3633**	0.3334
Proanthocyanins	0.3134	0.3046	0.1521
Nonflavanoid phenols	0.2985	0.2872	0.2453
Hue	0.2967	0.3090	0.2823
Proline	0.2868	0.3022	**0.3558**
Malic acid	0.2452	0.2237	0.2489
Alcalinity of ash	0.2393	0.2685	0.3481
Alcohol	0.1443	0.1886	0.3408
Magnesium	0.1420	0.1640	0.1399
Color intensity	0.0886	0.0550	0.0809
Ash	0.0021	0.0010	0.0893

Features	PCA	pairwise PCA	all PCA
Color intensity	**0.5300**	**0.5522**	**0.5696**
Alcohol	**0.4837**	**0.4429**	0.2506
Proline	**0.3649**	**0.3603**	0.2577
Ash	0.3161	0.3533	**0.3538**
Magnesium	0.2996	0.2679	**0.3946**
Hue	0.2792	0.2824	0.3146
Malic acid	0.2249	0.2240	0.0492
OD280/OD315 of diluted wines	0.1645	0.1697	0.2766
Total phenols	0.0650	0.0788	0.0717
Proanthocyanins	0.0393	0.0280	0.2065
Nonflavanoid phenols	0.0288	0.0769	0.0902
Alcalinity of ash	0.0106	0.0287	0.1363
Flavanoids	0.0034	0.0049	0.1147

Furthermore, Table 1.16 summaries the weight of each feature that the first and second principle components contain by PCA on full data and pairwise PCA and all PCA on data with missing values. The weights of the first three features with the highest weights are marked in bold. Since PCA is calculated from the full dataset, we want the results of pairwise PCA or all PCA to stay as close to the results of PCA as possible so that it captures more information, *i.e.* we want the first and second principal components in pairwise PCA or all PCA both assign

similar weights to each feature as PCA does. According to Table 1.16, the weights assigned by pairwise PCA and PCA are very close to each other. In contrast, while all PCA selects the same feature with highest weights as original PCA, the weights of other features it gives are generally much further away than pairwise PCA. We may conclude that pairwise PCA preserves as much information as possible from data with missing values and thus produces more similar results to PCA from the full dataset than all PCA. This indicates that pairwise PCA works better than all PCA on dataset with missing entries and PSDisation helps achieve this.

To further show the impact of missing values on the data and different methods of imputation data before PCA, we perform the following classification experiment. We split the dataset into 70% training set and 30% testing set. For the training set, we randomly drop 20% of the data so that the correlation matrix estimated pairwisely is not PSD. We use different PSDisation methods to obtain PSD correlation matrix and then perform PCA. In comparison, we use all instances without missing values to perform all PCA. We also include PCA using the correlation matrix estimated by the shrinkage method with the two data imputation strategies as explained previously in our test. As a benchmark, PCA is performed directly on data where the missing entries are replaced with mean values. For each of these PCA methods, the first 3 principle components are chosen to build a k nearest neighbours (kNN) model. We then use the kNN model to predict the labels with PCA-transformed data in the testing set and calculate the prediction accuracy. The test is repeated 50 times and average testing accuracies are shown in Table 1.17. In each model k is tuned using leave-one-out cross-validation.

Table 1.17: Average testing accuracy of kNN models built on data after PCA using different correlation matrix estimating methods. (Best performance is marked in bold.)

	Testing accuracy
all PCA	0.9143
pairwise PCA with ALD	0.9581
pairwise PCA with IQP	0.9581
pairwise PCA with ILP	**0.9589**
pairwise PCA with GD	0.9581
pairwise PCA with SDM	0.9581
PCA with Shrinkage-Mean	0.9555
PCA with Shrinkage-MICE	0.9585
PCA on data filled with mean values	0.9558

According to Table 1.17, all PCA performs the worst in the classification task, indicating that dropping too much data in the training set may lead to undesirable behaviours of the model. Pairwise PCA on the other hand gives better predictions, of which ILP achieves the best prediction accuracy. This shows that in some scenarios, optimising under the Chebyshev norm instead of the F-norm can practically be a better choice. The shrinkage method performs similarly to pairwise PCA and it also depends on data imputation methods. However, there is not enough evidence to judge between pairwise PCA and PCA with the shrinkage method.

1.7 Conclusions

In this chapter, we proposed two new approaches, the iterative quadratic/linear programming method and the gradient descent method, to tackle the PSDisation problem for actuarial analysis in order to find the nearest correlation matrix.

IQP/ILP is a flexible algorithm that approximates the nearest correlation matrix by solving a series of optimisations with linear constraints. Despite being more time-consuming than traditional methods, the IQP method can achieve optimal solutions that are comparable to the APM, Newton, or ALD method when working with the F-norm. There is currently no efficient PSDisation algorithm in literature that minimises the Chebyshev norm as ILP does. In our experiments, the ILP and IQP methods produce good results under the Chebyshev norm and the H-norm, respectively. In addition, it is crucial to understand that our formula-

tion of the problem is very flexible and thus can handle norms of any choice. It is also possible to combine different norms in our formulation (*e.g.* to minimise $\alpha||\mathbf{A}-\mathbf{X}||_{\mathbf{F}}^2+(1-\alpha)||\mathbf{A}-\mathbf{X}||_{\mathbf{C}}^2$ where α is a weighting parameter). Future work can be done to increase the speed of solving the quadratic or linear optimisation problem in each iteration to increase the efficiency in order to cope with larger problems.

The GD method repeats the following, taking a step in the opposite direction of the gradient, projecting the matrix onto the PSD cone via spectral decomposition and scaling the matrix so that the diagonal is 1. This method is effective in unconstrained PSDisation problems with respect to the F-norm. Experiments show the efficiency and robustness of the algorithm that it is more resilient to noise and shows good potential in practical scenarios. Future attempts to modify the GD algorithm can be considered so that it can be applied under different choices of norms or to solve problems with more constraints. For example, one possible extension of this could be to explore subgradient approach or use smooth approximation to work with norms that are not differentiable.

Chapter 2

Two Fair Regression Models with Liu-type Estimator

2.1 Introduction

Machine learning techniques have been developed rapidly over the past decades. They are widely used in decision making processes such as credit scoring, employment, sentence and bail decisions and play an important role in people's everyday life. On the one hand, machines can work with a much larger scale of datasets than human beings are able to and cost fewer time. On the other hand, it is commonly agreed that algorithms also suffer from biased as human beings do, even though there was no intention for it in the first place. A well-known example is Correctional Offender Management Profiling for Alternative Sanctions (COMPAS), a criminal justice scoring system to help the court to evaluate the chance of recommitting a crime for an offender and to decide whether to release him or keep in the prison in the US. Studies show that the false positive rate (FPR) for African-Americans is twice as that for white people. Therefore, concerns arise for eliminating bias and ensuring fairness in learning. [20, 33].

Biases in machine learning origin from different aspects. Firstly, the data used to feed the model might be biased, which will influence the algorithmic outcomes. An example for this is how we sample and measure the data. Data collected in a community which is highly policed and controlled may yield higher arrest rate

than other communities. Missing data or omitted variables also causes bias, as algorithms may fail to find the link between response and the missing variables but attribute to other irrelevant features. Imbalanced data also leads to unfairness, as optimising the overall error on the whole dataset may result in minority groups being ignored. Secondly, biases in algorithm can play an important role in causing people making unfair decisions. For example, a search engine provides links suggestions in correspondence to a search enquiry. However, the top few links are usually considered more relevant and are clicked by users more frequently, which can affect the performance of future searches. Lastly, user behaviour and decisions can in turn introduce bias to the data collection process. An example is that the most enthusiastic supporters of a political candidate are more likely to complete an opinion poll that measures enthusiasm then anyone else. What's more, data sampled on different social media platforms such as Facebook, Reddit or Twitter may be influenced by different user populations according to gender, age or ethnicity, etc. For a complete summary of sources of unfairness, see [34].

Starting a few years ago, there has been much interest in improving fairness while maintaining accuracy in prediction. In order to achieve fairness, we must first decide which features (groups) we would like to protect from prejudices when learning [35]. These are called sensitive features. These protected attributes are usually determined according to certain anti-discrimination laws, including gender, race, nationality, age, religion, etc. For example, we may want males and females to have equal opportunities to get a job, or black and white people to receive fair treatment in COMPAS.

Once the sensitive features are selected, the problem of how to define fairness comes up. Currently, most works on fairness are based on classification models [36, 37]. Popular fairness metrics include demographic parity, equalised odds and equalised opportunity [38]. Starting from here, there are also studies where they generalised those definitions into regression tasks, namely equalised expected outcomes and equalised expected log-likelihood [20]. [39] chose to measure fairness by the coefficient of determination (CoD) of sensitive attributes. We will further

introduce the fair learning problem and related definitions properly later in the next section. It is worth mentioning that different definitions may be useful in specific applications and no definition is guaranteed to work the best. Studies show that it is impossible to achieve fairness in multiple definitions at the same time unless with a trivial dataset [10]. Evidence shows that it is impossible either to achieve both predictive accuracy and model fairness at the same time (with a non-trivial dataset, of course). Indeed, consider the algorithm as an optimisation problem that aims to minimise the overall error which represents accuracy. Fairness can be achieved by adding additional constraints to the optimisation, leading to a narrower feasible region and therefore resulting a worse accuracy. [10] gave a more detailed discussion on trade-offs related to fairness.

In most literature researchers usually choose a definition that is most suitable for their model formulation. In this chapter, we will focus on linear regression tasks. Currently there are fewer studies focusing on regression than classification models. [19] used the Hilbert Schmidt independence criterion as the fairness term and applied their method to both linear and kernel regression. [17] proposed to use both individual fairness and group fairness and applied them to regression problems. [16] proposed a general scheme for fair regression with arbitrary Lipschitz loss functions. [39] proposed a fairness learning framework via a quadratically constrained quadratic program (QCQP) formulation. Both predictors and sensitive attributes can be continuous and multi-dimensional sensitive features are allowed. This ensures that it can be applied on a wide range of different types of datasets. Furthermore, the fairness is measured by CoD and can be directly controlled by user's input. [33] further extended this idea and presented a formulation of the optimisation with only a simple ridge penalty. This model is more efficient to solve, which has a solution that is partly in closed form. It can also be extended to different models or work with different penalties. [20] aimed at achieving equalised expected outcomes or equalised expected log-likelihoods rather that statistical parity. They proposed using a convex penalty term to achieve fairness. It is based on generalised linear models so allows different types of outcomes, but only with one

sensitive attribute. These limitations and non-flexibility motivates us to develop a simple model that can potentially work with different definitions of fairness and produce stable results.

Our idea was inspired by Liu-type estimator [40]. Liu estimator was proposed by [41] to tackle the multicollinearity problem in linear regression. It was further generalised into Liu-type estimator [40], which is shown to perform more stable than ridge regression under collinearity by reducing the bias. It allows a two-step adjustment of the model to minimise MSE by having a shrinkage parameter and a correction parameter to reduce the condition number and improve the fitness. In this chapter, we use the parameters to balance the trade-off between fairness and accuracy and obtain robust linear models. We proposed two simple Liu-type linear models that achieves better fairness by selecting optimal parameters. The optimisation can be done by simply solving a linear or quadratic equation, which makes the model extremely easy to fit. Experiments show that they could achieve competitive trade-offs between accuracy and fairness to state-of-the-art methods while enjoying good stability.

This chapter is organised as follows. In Section 2.2 we formally introduce the fairness problem in machine learning by giving some notations and definitions of fairness. We start from the classification problem and then show how the ideas can be transferred into regression cases. Related fair learning methods and background knowledge is also introduced. In Section 2.3 we proposed two linear models, namely fair Liu model (fLiu) and fair Liu model with sensitive features (fLiuS). Experimental results for both models are reported and discussed in Section 2.4. Finally, in Section 2.5 we conclude this chapter with some future work ideas.

2.2 Related Work

2.2.1 Fairness in Classification Tasks

The concept of fair learning origins from binary classification tasks with one single sensitive feature so we will start to introduce the definitions from here. Let A denote a potentially sensitive attribute that we wish to protect which takes discrete values

from the set $\mathscr{A} = \{a_1, \ldots, a_k\}$. Let $\boldsymbol{X} \in \mathbb{R}^{d_x}$ denote other attributes which can be either discrete or continuous and d_x is the dimension of the features. Denote the outcome group by Y which for now we assume is binary, *i.e.* $\mathscr{Y} = \{0, 1\}$, and denote the binary predictor by $\hat{Y}$. However, many of the definitions can be generalised to work with categorical outcomes.

An intuitive way for attribute protection is by fairness through unawareness, where the protected attributes A are simply left out of the model.

Definition 1 (Fairness through unawareness)**.** *A predictor $\hat{Y}$ satisfies fairness through unawareness if the sensitive attribute A is not used in the predictive model.*

However, this may have limited effect on addressing and eliminating bias since the protected attributes may be correlated to some attributes in $\boldsymbol{X}$ [42], resulting the model to be still biased. Furthermore, ensuring independence of $\hat{Y}$ and A would result in bad predictions when the sensitive feature A itself can be a good predictor of the response Y.

Definition 2 (Fairness through awareness)**.** *A predictor $\hat{Y}$ satisfies fairness through awareness if each pair of two similar instances receives similar treatment in the predictive model.*

Both Definitions 1 and 2 are fairness on an individual level. Definition 2 requires a more specific definition on similarity (distance metric) between pair of instances, which is non-trivial and must be designed according to the intended tasks.

Further criteria for group fairness have been created in order to treat different groups equally rather than looking at individual instances.

Definition 3 (Independence/demographic parity)**.** *A predictor $\hat{Y}$ satisfies independence from sensitive attribute A if $P(\hat{Y} = 1 \mid A = a_i) = P(\hat{Y} = 1 \mid A = a_j)\ \forall\ 1 \leq i, j \leq k$.*

Independence of score and sensitive attribute requires $\hat{Y}$ and A to be statistically independent, *i.e.* $\hat{Y} \perp A$. This requires the learning algorithm to be consistent in the entire dataset so that it does not distinguish instances from different sensitive

groups, which potentially contributes to its drawback since it prevents us from utilising knowledge we could learn on each separate class [36]. For example, a perfect predictive model may be judged as unfair under this definition if the actual positive rates of different sensitive groups are different [10], which is often the case in realistic. Moreover, on an individual fairness perspective, two similar individuals from different sensitive groups can be treated differently, which in some scenario can be against the law.

Definition 4 (Separation/equalised odds)**.** *A predictor $\hat{Y}$ satisfies separation with respect to sensitive attribute A given outcome Y if* $P(\hat{Y} = 1 \mid A = a_i, Y = y) = P(\hat{Y} = 1 \mid A = a_j, Y = y)\ \forall\ 1 \leq i, j \leq k,\ y \in \{0,1\}$.

Separation requires that the protected attribute is only related to the score through the outcome, *i.e.* $\hat{Y} \perp A \mid Y$ [36]. Therefore the sensitive attribute can be correlated with the predictor and for different sensitive groups we may have different rates of positive predictions based on outcomes in training data.

Definition 5 (Equalised opportunity)**.** *A predictor $\hat{Y}$ satisfies equalised opportunity with respect to sensitive attribute A given outcome Y if* $P(\hat{Y} = 1 \mid A = a_i, Y = 1) = P(\hat{Y} = 1 \mid A = a_j, Y = 1)\ \forall\ 1 \leq i, j \leq k$.

Equalised opportunity is a weaker fairness condition than equalised odds where only the instances predicted to be accepted are concerned to be fairly scored while others are ignored [36]. Depending on the applications equalised opportunity can be easier to achieve than equalised odds.

Definitions 4 and 5 are two popular definitions for classification fairness based on the most current studies, though equalised odds is still proven to suffer from difference in false negative rate and equalised opportunity may behave badly when base rates from different sensitive groups differ [10].

Definition 6 (Sufficiency)**.** *A predictor $\hat{Y}$ satisfies sufficiency with respect to sensitive attribute A and outcome Y if* $P(Y = 1 \mid A = a_i, \hat{Y} = y) = P(Y = 1 \mid A = a_j, \hat{Y} = y)\ \forall\ 1 \leq i, j \leq k,\ y \in \{0,1\}$, *i.e.* $Y \perp A \mid \hat{Y}$.

Above are examples of definitions for fairness in classification (binary) tasks. We refer the readers to [34, 10, 43] for more detailed discussions on this topic.

2.2.2 Fairness in Regression Tasks

In this section, we introduce fairness in regression tasks. We mainly introduce two representative algorithms, fair generalised linear models (FGLM) and fair ridge regression model (FRRM), which we consider to be state-of-the-art. The formulation of the problem is very similar to that in classification, while in regression tasks we allow the outcome Y and the predictive score $\hat{Y}$ to have real values. Many definitions we discussed previously can be adapted directly into regression problems. Here we give an example of the generalised version of Definition 4 as follows [20].

Definition 7 (Equalised expected outcomes)**.** *A predictive score $\hat{Y}$ satisfies equalised expected outcomes with respect to sensitive attribute A given outcome Y if $\mathbb{E}(\hat{Y} \mid A = a_i, Y = y) = \mathbb{E}(\hat{Y} \mid A = a_j, Y = y)\ \forall\ 1 \leq i, j \leq k,\ y \in \mathscr{Y}$.*

Since Y is real-valued here, we discritise it into a finite number of regions with segments that are closed on the left and open on the right to form $\mathscr{Y}$.

Before going through the methods, we first declare the notations we use in the fair regression problem. Let $\boldsymbol{S} \in \mathbb{R}^{d_s}$ be the sensitive variables and $\boldsymbol{X} \in \mathbb{R}^{d_x}$ be the normal (non-sensitive) variables where d_s and d_x are the dimensions of the sensitive and non-sensitive features, respectively. Denote the continuous outcome variable by $Y \in \mathbb{R}$. Let $\mathbf{S} \in \mathbb{R}^{n \times d_s}$ be observations of the sensitive attributes and $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ be observations of the non-sensitive attributes and n is the number of observations. Let $\mathbf{Y} \in \mathbb{R}^{n \times 1}$ be the target attribute to predict. We assume that all categorical variables are one-hot encoded. Let $\boldsymbol{\beta} \in \mathbb{R}^{d_x}$ be the coefficient vector of a linear regression model (assuming that the sensitive features are not included in the model) and $\mathbf{X}\boldsymbol{\beta}$ is the linear predictor.

2.2.2.1 Fair Generalised Linear Model

[20] proposed a unified framework for fair GLM estimator (FGLM). The idea is to solve an optimisation problem that minimises negative log-likelihood plus a penalty term weighted by a constant λ that controls the trade-off between accuracy and

fairness. The penalty term is determined as a measure of fairness, whose value is the lower the better. They built their FGLM algorithm based on two fairness criteria, equalised expected outcomes and equalised expected log-likelihood. In their framework, they assume that the sensitive features **S** contain only one multi-class variable S. Let $p^i_{\boldsymbol{X}|Y=y}$ represents the conditional distribution of $\boldsymbol{X}$ given that $Y = y$ and $S = s_i$, $i = 1,\ldots,k$. The full FGLM formulations will be discussed in Chapter 3, as the main focus of this chapter is on linear regression. For the linear regression case (with the identity link function), the equation in Definition 7 can be written as,

$$\mathbb{E}(\boldsymbol{X}^{iy}\boldsymbol{\beta}) = \mathbb{E}(\boldsymbol{X}^{jy}\boldsymbol{\beta}) \;\forall\; 1 \leq i,j \leq k,\; y \in \mathscr{Y},$$

where $\boldsymbol{X}^{iy} \sim p^i_{\boldsymbol{X}|Y=y}$ and $\boldsymbol{X}^{jy} \sim p^j_{\boldsymbol{X}|Y=y}$. Note that in a linear model the expected predictive score equals to the linear predictor. Based on this definition, [20] defined a measure of group fairness by summing up the squared differences of the pairwise expected outcomes $\mathscr{D}_{EO}$,

$$\mathscr{D}_{EO} = \sum_{i,j=1}^{k} \sum_{y\in\mathscr{Y}} (\mathbb{E}(\boldsymbol{X}^{iy}\boldsymbol{\beta}) - \mathbb{E}(\boldsymbol{X}^{jy}\boldsymbol{\beta}))^2.$$

where again we discritise the range of real-valued Y into a finite number of regions with segments that are closed on the left and open on the right to form $\mathscr{Y}$.

The fairness measured by the expected squared difference of the pairwise expected log-likelihood $\mathscr{D}_{ELL}$ can be defined in a similar way.

In order to achieve efficiency in computation, a new measure of fairness $\mathscr{D}_{LC}$ is created and used in optimisation. $\mathscr{D}_{LC}$ is defined by summing up the expected squared differences of each pair of linear components from different sensitive class,

$$\mathscr{D}_{LC} = \sum_{i,j=1}^{k} \sum_{y\in\mathscr{Y}} \mathbb{E}((\boldsymbol{X}^{iy}\boldsymbol{\beta} - \boldsymbol{X}^{jy}\boldsymbol{\beta})^2). \tag{2.1}$$

With the above definition of fairness, [20] proposed to find their fair GLM

estimator by minimising the following function over $\boldsymbol{\beta}$,

$$-\mathbb{E}(\ell(\boldsymbol{\beta};\boldsymbol{X},Y))+\frac{\lambda}{\kappa}\sum_{i,j=1}^{k}\sum_{y\in\mathscr{Y}}\mathbb{E}((\boldsymbol{X}^{iy}\boldsymbol{\beta}-\boldsymbol{X}^{jy}\boldsymbol{\beta})^2),$$

where $\ell(\boldsymbol{\beta};\boldsymbol{X},Y)$ is the log-likelihood of the GLM, $\kappa=\frac{k(k-1)}{2}|\mathscr{Y}|$ is the number of possible combinations and $\lambda>0$ is the weighting parameter which controls the trade-off between fairness and predictive accuracy. By using $\mathscr{D}_{LC}$ as the penalty term which can be simplified into matrix form, the optimisation is convex and therefore can be solved efficiently through a Newton-Raphson method. [20] also showed that by changing the weighting parameter λ, both $\mathscr{D}_{EO}$ and $\mathscr{D}_{ELL}$ can be effectively bounded.

FGLM turns out to work efficiently with GLM so it can handle different types of tasks including binary and multi-class classifications, continuous and count outcomes regressions. However, continuous sensitive attribute or multi-dimensional sensitive attributes are not supported. In addition, since the response Y is continuous, a discretisation is needed before we can calculate the fairness penalty. Therefore the overall performance may vary depending on different discretisation method. It is also worth pointing out that the sensitive feature is not used to feed the model, but only used to calculate the fairness penalty.

2.2.2.2 Fair Ridge Regression Model

Next we introduce the fair ridge regression model (FRRM). It was proposed by [33] which was adapted from the formulations of [44] and [39] to overcome the non-convexity of the model from earlier works. A major difference from FGLM is that sensitive features $\mathbf{S}$ are allowed to be multi-dimensional and will be used in model prediction. The model is designed and the fairness is measured as follows.

Firstly a multivariate linear regression is done to remove the correlation between $\mathbf{X}$ and $\mathbf{S}$,

$$\mathbf{X}=\mathbf{S}\mathbf{B}^T+\mathbf{U},$$

where $\mathbf{B}$ is the regression coefficients and $\mathbf{U}$ is the residuals. $\mathbf{B}$ is estimated by ordinary least squares as $\hat{\mathbf{B}}_{OLS}$ and then the residuals are given by,

$$\hat{\mathbf{U}} = \mathbf{X} - \mathbf{S}\hat{\mathbf{B}}_{OLS}^{T}.$$

where $\mathbf{S}$ and $\hat{\mathbf{U}}$ are orthogonal and $\mathrm{COV}(\mathbf{S}, \hat{\mathbf{U}}) = \mathbf{0}$.

Then the predictive model is fitted with $\mathbf{S}$ and $\hat{\mathbf{U}}$,

$$\mathbf{Y} = \mathbf{S}\boldsymbol{\alpha} + \hat{\mathbf{U}}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{d_s \times 1}$ and $\boldsymbol{\beta} \in \mathbb{R}^{d_x \times 1}$. The fairness can be determined by the overall explained variance that is attributeable to the ensitive attributes $R^2_{\mathbf{S}}$ (or coefficient of determination, CoD, of the sensitive attributes),

$$\mathrm{CoD} = \frac{\mathrm{Var}\,(\mathbf{S}\boldsymbol{\alpha})}{\mathrm{Var}\,(\hat{\mathbf{Y}})} = \frac{\boldsymbol{\alpha}^T \mathbf{V}_{\mathbf{S}} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{V}_{\mathbf{S}} \boldsymbol{\alpha} + \boldsymbol{\beta}^T \mathbf{V}_{\mathbf{U}} \boldsymbol{\beta}}, \tag{2.2}$$

where $\mathbf{V}_{\mathbf{S}} = \mathrm{COV}\,(\mathbf{S})$ and $\mathbf{V}_{\mathbf{U}} = \mathrm{COV}\,(\hat{\mathbf{U}})$.

Based on the above definition of CoD, [33] achieves fairness by minimising the following objective function over $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$,

$$||\mathbf{Y} - \mathbf{S}\boldsymbol{\alpha} - \hat{\mathbf{U}}\boldsymbol{\beta}||_2^2 + \lambda(r)||\boldsymbol{\alpha}||_2^2,$$

where $\lambda(r) \geq 0$ is a parameter such that $\mathrm{CoD}(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leq r$.

The objective function consists of two parts, the mean squared error plus a simple ridge penalty on $\boldsymbol{\alpha}$ only. This penalty is parameterised such that CoD can be bounded at an arbitrary threshold r. It can be further shown that there is a closed-form solution to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ with respect to the weighting parameter for the ridge penalty, which monotonically controls CoD. Therefore the optimisation can be done through a simple root-finding algorithm.

FRRM is easy to implement and can work with continuous or multi-dimensional sensitive attributes. Sensitive features are used both as inputs of the model and also to calculate fairness. However, ground truth $\mathbf{Y}$ is not included in the

definition of fairness CoD.

Table 2.1 summarises ideas and differences between these two definitions of fairness used in FRRM and FGLM.

fairness	CoD	$\mathscr{D}_{LC}$
Definition	Proportion of the overall explained variance that is attributable to the sensitive attributes	Sum of squared differences of pairwise linear components $X\beta$
Idea	Statistical parity (predictions and the sensitive variables are independent)	Equalised expected outcomes (add a variance term to bound also difference of expected log-likelihoods)
Use of ground truth	No	Yes

Table 2.1: Comparisons of two definitions for regression fairness.

2.3 Methodology

In this chapter, we propose our two approaches to achieve fairness in linear regression models that are efficient and robust. Our idea is inspired by Liu-type estimator [40] which has been shown to be able to work well with the presence of collinearity. It simply uses two parameters to addresses the ill conditioning problem with a closed-form solution. Therefore, depending on different model settings or definitions of fairness, we propose fair Liu model (fLiu) and fair Liu model with sensitive features (fLiuS). The first model fLiu is built using only nonsensitive features $\mathbf{X}$ and parameter d can be calculated to minimise the expected squared difference of linear components $\mathscr{D}_{LC}$ in (2.1). The second model fLiuS takes both $\mathbf{S}$ and $\hat{\mathbf{U}}$ as inputs and aims to minimises CoD in (2.2). By alternating the parameters, we can achieve a trade-off between fairness and accuracy effectively. Furthermore, depending on the choice of k, the optimisation of $\mathscr{D}_{LC}$ or CoD can be achieved within the Liu-type framework by choosing parameter value d simply by solving a linear or quadratic equation, which makes the model extremely easy to fit. We will first introduce Liu-type estimator, followed by our two proposed methods.

2.3.1 Liu-type Estimator

Liu-type estimator [40] can be seen as a generalisation of Liu estimator [41] used to combat collinearity. Before that, a well-known method was ridge regression

proposed by [45],

$$\hat{\boldsymbol{\beta}}_R = (\mathbf{X}^T\mathbf{X} + k\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}.$$

The shrinkage parameter $k > 0$ is used to control the condition number of $\mathbf{X}^T\mathbf{X} + k\mathbf{I}$, as a large condition number will cause the estimator to be unstable. However, in application, by increasing k in order to reduce the condition number will in turn introduce more biasd to the regression [40]. Liu-type estimator was then proposed to overcome this problem,

$$\hat{\boldsymbol{\beta}}_{k,d} = (\mathbf{X}^T\mathbf{X} + k\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{Y} + d\hat{\boldsymbol{\beta}}),$$

where $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ is the least squares estimator. $-\infty < d < \infty$ is to ensure that the model gets a good fit and is called the correction parameter.

The Liu-type estimator can be seen as a generalisation of Stein's shrinking principle [46], that is, to improve the estimator by shrinking an unbiased estimator toward a target with lower variance to reduce MSE. It helps improve overall estimation by trading off bias and variance and works effectively in high-dimensional data with ill-conditioned settings. The Liu-type estimator can be interpreted as a combination of two estimators, typically the ordinary least squares (OLS) estimator and a structured prior. The shrinkage parameter k controls the shrinkage behaviour of the estimator and helps combat multicollinearity. A large k shrinks the model more and improves the stability of the model, as in ridge regression, but introduces more bias. The correction parameter d controls the bias towards structured shrinkage and helps retain the OLS solution. From this perspective, the Liu-type estimator offers a flexible framework that benefits from the advantages of both ridge regression and the original Stein estimator, and can be adapted to fair learning according to different fairness constraints or penalties. Instead of the commonly used OLS estimator $\hat{\boldsymbol{\beta}}$ in the formulation, we can also choose $\hat{\boldsymbol{\beta}}_R$. This may lead to a more robust model, but it also adds complexity by introducing one more parameter, and the performance depends on specific applications. We will consider using the OLS estimator $\hat{\boldsymbol{\beta}}$ in

this chapter.

[40] showed that for any $k > 0$, there exists d such that Liu-type estimator gives less or equal MSE than ridge estimator. Our work will start from Liu-type estimator and try to achieve optimal fairness by selecting d given k. We use the two fairness metrics in FGLM and FRRM, $\mathscr{D}_{LC}$ and CoD, to demonstrate the effectiveness of utilising d in Liu-type estimator for achieving fairness.

2.3.2 Fair Liu Model

We minimise $\mathscr{D}_{LC}$ by choosing d in our first model Liu, which is fitted without sensitive features. The Liu-type estimator in this model $\hat{\boldsymbol{\beta}}_{k,d} = (\mathbf{X}^T\mathbf{X}+k\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X}+d\mathbf{I})\hat{\boldsymbol{\beta}}$ where $\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$. Note that here $\mathbf{X}$ does not involve sensitive features.

The fairness measure $\mathscr{D}_{LC}$ can be further written as,

$$
\begin{aligned}
\mathscr{D}_{LC} &= \frac{1}{\kappa}\sum_{k,l\in\mathscr{A}}\sum_{y\in\mathscr{Y}}\mathbb{E}[(\mathbf{X}^{ky}\hat{\boldsymbol{\beta}}_{k,d}-\mathbf{X}^{ly}\hat{\boldsymbol{\beta}}_{k,d})^2] \\
&\approx \frac{1}{\kappa}\sum_{k,l\in\mathscr{A}}\sum_{y\in\mathscr{Y}}\frac{1}{n^{kly}}\sum_{(i,j)\in S^{kly}}(\boldsymbol{x}_i\hat{\boldsymbol{\beta}}_{k,d}-\boldsymbol{x}_j\hat{\boldsymbol{\beta}}_{k,d})^2 \\
&= \hat{\boldsymbol{\beta}}_{k,d}^T\underbrace{[\frac{1}{\kappa}\sum_{k,l\in\mathscr{A}}\sum_{y\in\mathscr{Y}}\frac{1}{n^{kly}}\sum_{(i,j)\in S^{kly}}(\boldsymbol{x}_i-\boldsymbol{x}_j)^T(\boldsymbol{x}_i-\boldsymbol{x}_j)]}_{\mathbf{D}}\hat{\boldsymbol{\beta}}_{k,d} \\
&= \hat{\boldsymbol{\beta}}_{k,d}^T\mathbf{D}\hat{\boldsymbol{\beta}}_{k,d} \\
&= [(\mathbf{X}^T\mathbf{X}+k\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X}+d\mathbf{I})\hat{\boldsymbol{\beta}}]^T\mathbf{D}[(\mathbf{X}^T\mathbf{X}+k\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{X}+d\mathbf{I})\hat{\boldsymbol{\beta}}] \\
&= \hat{\boldsymbol{\beta}}^T(\mathbf{X}^T\mathbf{X}+d\mathbf{I})\underbrace{(\mathbf{X}^T\mathbf{X}+k\mathbf{I})^{-1}\mathbf{D}(\mathbf{X}^T\mathbf{X}+k\mathbf{I})^{-1}}_{\mathbf{A}_k}(\mathbf{X}^T\mathbf{X}+d\mathbf{I})\hat{\boldsymbol{\beta}} \\
&= \hat{\boldsymbol{\beta}}^T(\mathbf{X}^T\mathbf{X}\mathbf{A}_k\mathbf{X}^T\mathbf{X}+d\mathbf{A}_k\mathbf{X}^T\mathbf{X}+\mathbf{X}^T\mathbf{X}\mathbf{A}_k d+d^2\mathbf{A}_k)\hat{\boldsymbol{\beta}} \\
&= \hat{\boldsymbol{\beta}}^T\mathbf{A}_k\hat{\boldsymbol{\beta}}d^2+(\hat{\boldsymbol{\beta}}^T\mathbf{A}_k\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}+\hat{\boldsymbol{\beta}}^T\mathbf{X}^T\mathbf{X}\mathbf{A}_k\hat{\boldsymbol{\beta}})d+C,
\end{aligned}
$$

where C is some term that does not involve d. This is a quadratic function of d and

therefore is minimised at,

$$d_{fair} = -\frac{\hat{\boldsymbol{\beta}}^T \mathbf{A}_k \mathbf{X}^T \mathbf{X} \hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\beta}}^T \mathbf{X}^T \mathbf{X} \mathbf{A}_k \hat{\boldsymbol{\beta}}}{2\hat{\boldsymbol{\beta}}^T \mathbf{A}_k \hat{\boldsymbol{\beta}}}.$$

This gives us a fLiu estimator that depends on one parameter k only.

2.3.3 Fair Liu Model with Sensitive Features

We minimise CoD by choosing d in our second model fLiuS. Denote $\mathbf{W} = [\mathbf{S}, \hat{\mathbf{U}}] \in \mathbb{R}^{n \times (d_s + d_x)}$. Suppose we have a Liu-type estimator $\boldsymbol{\beta}_{k,d} = [\boldsymbol{\alpha}, \boldsymbol{\beta}]^T \in \mathbb{R}^{(d_s+d_x)\times 1}$ which is estimated using $\mathbf{W}$. Then,

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{W}\boldsymbol{\beta}_{k,d}, \\ &= \mathbf{S}\boldsymbol{\alpha} + \hat{\mathbf{U}}\boldsymbol{\beta} \end{aligned}$$

where $\hat{\boldsymbol{\beta}_{k,d}} = (\mathbf{W}^T\mathbf{W} + k\mathbf{I})^{-1}(\mathbf{W}^T\mathbf{W} + d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}$ and $\hat{\boldsymbol{\beta}}_{LS} = (\mathbf{W}^T\mathbf{W})^{-1}\mathbf{W}^T\mathbf{Y}$.

The coefficient of determination can be given by,

$$\text{CoD} = \frac{\text{Var}(\mathbf{S}\boldsymbol{\alpha})}{\text{Var}(\hat{\mathbf{Y}})} = \frac{\boldsymbol{\beta}_{k,d}^T \mathbf{V}_{\mathbf{SI}} \boldsymbol{\beta}_{k,d}}{\boldsymbol{\beta}_{k,d}^T \mathbf{V}_{\mathbf{W}} \boldsymbol{\beta}_{k,d}},$$

where $\mathbf{V}_{\mathbf{SI}} = \text{COV}(\mathbf{S}\mathbf{I}_\alpha)$, $\mathbf{V}_{\mathbf{W}} = \text{COV}(\mathbf{W})$ and $\mathbf{I}_\alpha$ is a $d_s \times (d_s + d_x)$ matrix with the following form,

$$\mathbf{I}_\alpha = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}.$$

We could further write CoD as a function of d,

$$\begin{aligned}
\text{CoD} &= \frac{\boldsymbol{\beta}_{k,d}^T \mathbf{V_{SI}} \boldsymbol{\beta}_{k,d}}{\boldsymbol{\beta}_{k,d}^T \mathbf{V_W} \boldsymbol{\beta}_{k,d}} \\
&= \frac{[(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}(\mathbf{W}^T\mathbf{W}+d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}]^T \mathbf{V_{SI}} [(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}(\mathbf{W}^T\mathbf{W}+d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}]}{[(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}(\mathbf{W}^T\mathbf{W}+d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}]^T \mathbf{V_W} [(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}(\mathbf{W}^T\mathbf{W}+d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}]} \\
&= \frac{\hat{\boldsymbol{\beta}}_{LS}^T(\mathbf{W}^T\mathbf{W}+d\mathbf{I}) \overbrace{(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}\mathbf{V_{SI}}(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}}^{\mathbf{A_{SI}}} (\mathbf{W}^T\mathbf{W}+d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}}{\hat{\boldsymbol{\beta}}_{LS}^T(\mathbf{W}^T\mathbf{W}+d\mathbf{I}) \underbrace{(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}\mathbf{V_W}(\mathbf{W}^T\mathbf{W}+k\mathbf{I})^{-1}}_{\mathbf{A_W}} (\mathbf{W}^T\mathbf{W}+d\mathbf{I})\hat{\boldsymbol{\beta}}_{LS}} \\
&= \frac{\hat{\boldsymbol{\beta}}_{LS}^T\mathbf{A_{SI}}\hat{\boldsymbol{\beta}}_{LS}d^2 + (\hat{\boldsymbol{\beta}}_{LS}^T\mathbf{A_{SI}}\mathbf{W}^T\mathbf{W}\hat{\boldsymbol{\beta}}_{LS} + \hat{\boldsymbol{\beta}}_{LS}^T\mathbf{W}^T\mathbf{W}\mathbf{A_{SI}}\hat{\boldsymbol{\beta}}_{LS})d + \hat{\boldsymbol{\beta}}_{LS}^T\mathbf{W}^T\mathbf{W}\mathbf{A_{SI}}\mathbf{W}^T\mathbf{W}\hat{\boldsymbol{\beta}}_{LS}}{\hat{\boldsymbol{\beta}}_{LS}^T\mathbf{A_W}\hat{\boldsymbol{\beta}}_{LS}d^2 + (\hat{\boldsymbol{\beta}}_{LS}^T\mathbf{A_W}\mathbf{W}^T\mathbf{W}\hat{\boldsymbol{\beta}}_{LS} + \hat{\boldsymbol{\beta}}_{LS}^T\mathbf{W}^T\mathbf{W}\mathbf{A_W}\hat{\boldsymbol{\beta}}_{LS})d + \hat{\boldsymbol{\beta}}_{LS}^T\mathbf{W}^T\mathbf{W}\mathbf{A_W}\mathbf{W}^T\mathbf{W}\hat{\boldsymbol{\beta}}_{LS}}
\end{aligned}$$

By further differentiating with respect to d, we could find the d value that minimises CoD, which defines our fLiuS estimator with a single parameter k.

In Table 2.2 we summarise the ideas and properties of fLiu and fLiuS compared with state-of-the-art methods FGLM and FRRM.

	FGLM	fLiu	FRRM	fLiuS
Method	Maximise log-likelihood with a $\mathscr{D}_{LC}$ penalty	Linear regression by Liu-type estimator using only non-sensitive features, where d can be chosen to minimise $\mathscr{D}_{LC}$	Minimise sum of the squares of the residuals with a ridge penalty (for sensitive terms only), whose parameter is determined to bound the CoD	Linear regression by Liu-type estimator using both sensitive and non-sensitive features, where d can be chosen to minimise CoD
Use of sensitive features **S** in the prediction model	No	No	Yes	Yes
Fairness used for penalty (trying to reduce)	$\mathscr{D}_{LC}$	$\mathscr{D}_{LC}$	CoD	CoD
Allow multi-dimension of **S**	No	No	Yes	Yes
Allow continuous **S**	No	No	Yes	Yes
Allow continuous outcome **Y**	Yes	Yes	Yes	Yes
Allow other types (binary, count, multiclass) of outcomes **Y**	Yes	No	No	No

Table 2.2: Summary and comparisons between FGLM, fLiu, FRRM and fLiuS.

2.4 Experimental Results

In this section, we will evaluate the performance of our models, fLiu and fLiuS, in simulated data and real data. We will also verify their robustness.

2.4.1 Experiments on DAG Simulated Data

2.4.1.1 Simulated Data

We first create simulated data for our experiments. We generate the data according to the following directed acyclic graph (DAG) and distributions.

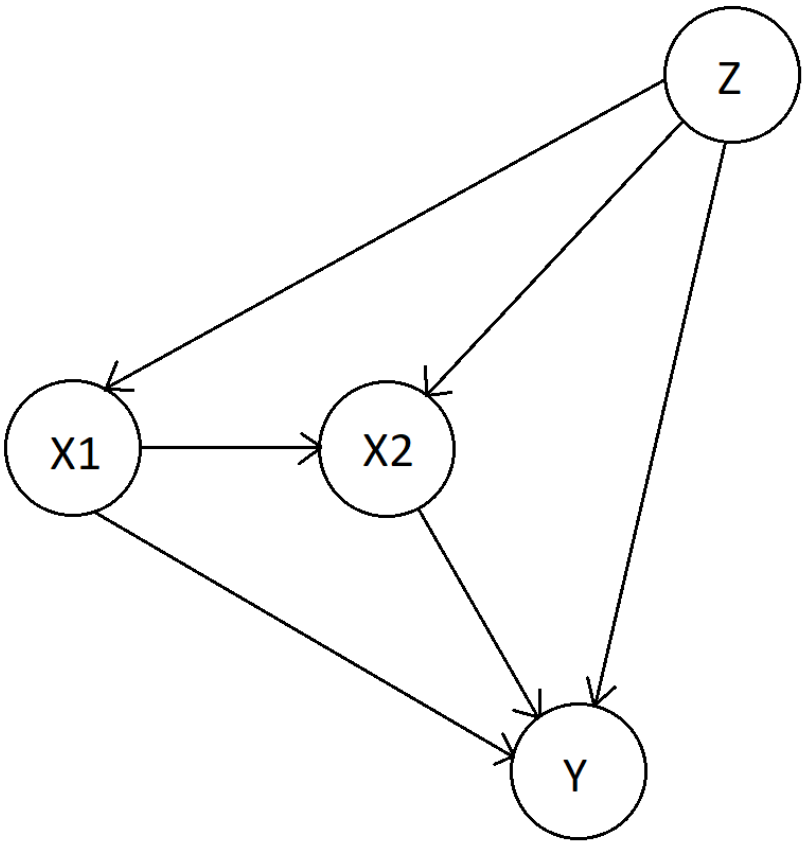

Figure 2.1: DAG used to simulate the dataset.

$$
\begin{aligned}
Z &\sim Bernoulli(0.5) \\
X_1 &= 4 * Z + \mathcal{N}(50, 4^2) \\
X_2 &= 12 + 1.1 * Z + 0.4 * X_1 + \mathcal{N}(0, 2^2) \\
Y &= -5 + 0.1 * X_1 + 0.3 * X_2 + h * Z + \mathcal{N}(0, 1^2)
\end{aligned}
$$

In this graph, X_1, X_2 are non-sensitive features and Y is the response. We consider Z to be the sensitive feature, where $Z = 1$ stands for female and $Z = 0$ represents the male group. Z has equal chances to be 0 or 1 so we consider the dataset to be balanced. We use parameter h to modify the expected values of Y in the following way. If $h = 0.2$, X_1, X_2 and Y all have larger expected values when $Z = 1$ than $Z = 0$. While if $h = -1.21$, $\mathbb{E}(Y|Z = 1) = \mathbb{E}(Y|Z = 0)$, though X_1 and X_2 still have different expectations.

2.4.1.2 Experimental Settings

We first want to show the trade-offs of prediction accuracy and fairness of different models. This is shown by varying the value of parameters and creating a plot of prediction accuracy versus fairness for each model. The models we consider here include fair generalised linear models (FGLM), fair ridge regression model (FRRM) and linear regression model using Liu-type estimator (fLiu/fLiuS), with and without sensitive features, respectively. FGLM contains one tuning parameter $\lambda \geq 0$ which controls the trade-off between fairness and log-likelihood. FRRM has one parameter $r \in [0, 1]$ that bounds the coefficient of determination, CoD, of the sensitive attributes, determining how fair the model is. In both fLiu and fLiuS we have two parameters k and d. The shrinkage parameter $k \geq 0$ can be used control the condition number of $\mathbf{X}^T\mathbf{X} + k\mathbf{I}$. However, in this experiment we fix it to 1, as varying it turns out to have little impact on the results. Parameter d can then be chosen via solving a linear or quadratic equation to achieve optimal fairness defined by $\mathscr{D}_{LC}$ in fLiu or CoD in fLiuS.

The data is split randomly into training (70%) and testing set (30%). First

we compare our method fLiu with FGLM, both of which use $\mathscr{D}_{LC}$ as the fairness metric. Both methods take only $\mathbf{X}$ as inputs to the model but does not include sensitive features Z. Instead, Z is used to calculate the fairness penalty during training. The accuracy is determined by root mean square error (RMSE). We calculate both $\mathscr{D}_{LC}$ and $\mathscr{D}_{EO}$ as there is no clear evidence which one works better. We try to fit the models with different hyperparameter λ for FGLM and d for fLiu and plot the curves to show the trade-offs between accuracy and fairness for each model in Figures 2.2 and 2.3. We also compare our method fLiuS with FRRM, both of which use sensitive features in the model and CoD as the fairness metric. Again the parameter k for fLiuS is fixed to 1. The trade-offs between accuracy and fairness is shown in Figure 2.4. Both experiments are repeated 20 times and average results are reported. We also record the computational times which will be reported later in this chapter.

2.4.1.3 Experimental Results

Each figure shows results for training (on the left) and testing datasets (on the right) generated using different h values. The top row is for $h = 0.2$, where the two sensitive groups have the most significant differences, while the bottom row is for $h = -1.21$, where $\mathbb{E}(Y|Z = 1) = \mathbb{E}(Y|Z = 0)$.

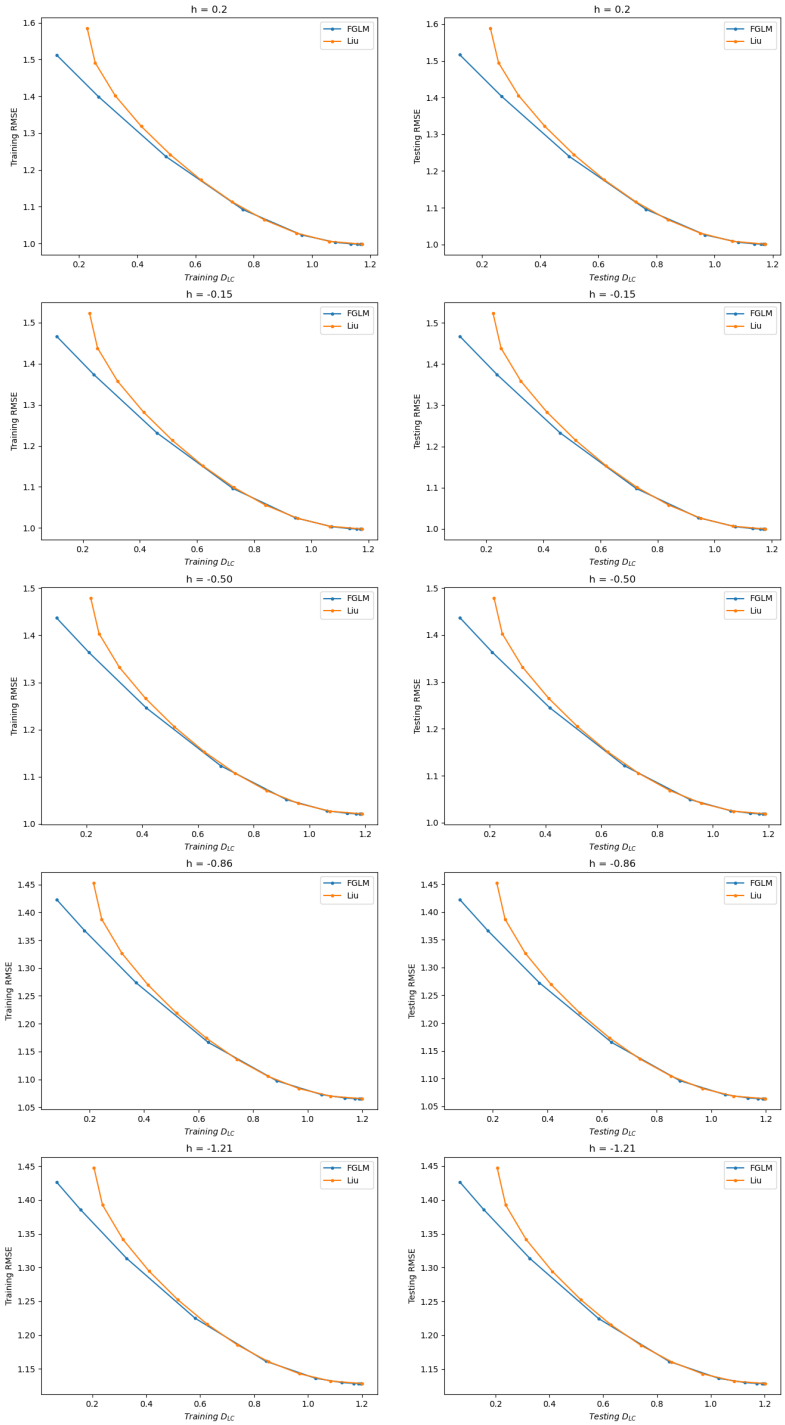

Figure 2.2: Experimental results on the DAG simulated dataset generated with different h values for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$) on training (left) and testing (right) sets of different linear models fitted without sensitive features. Both values are the lower the better.

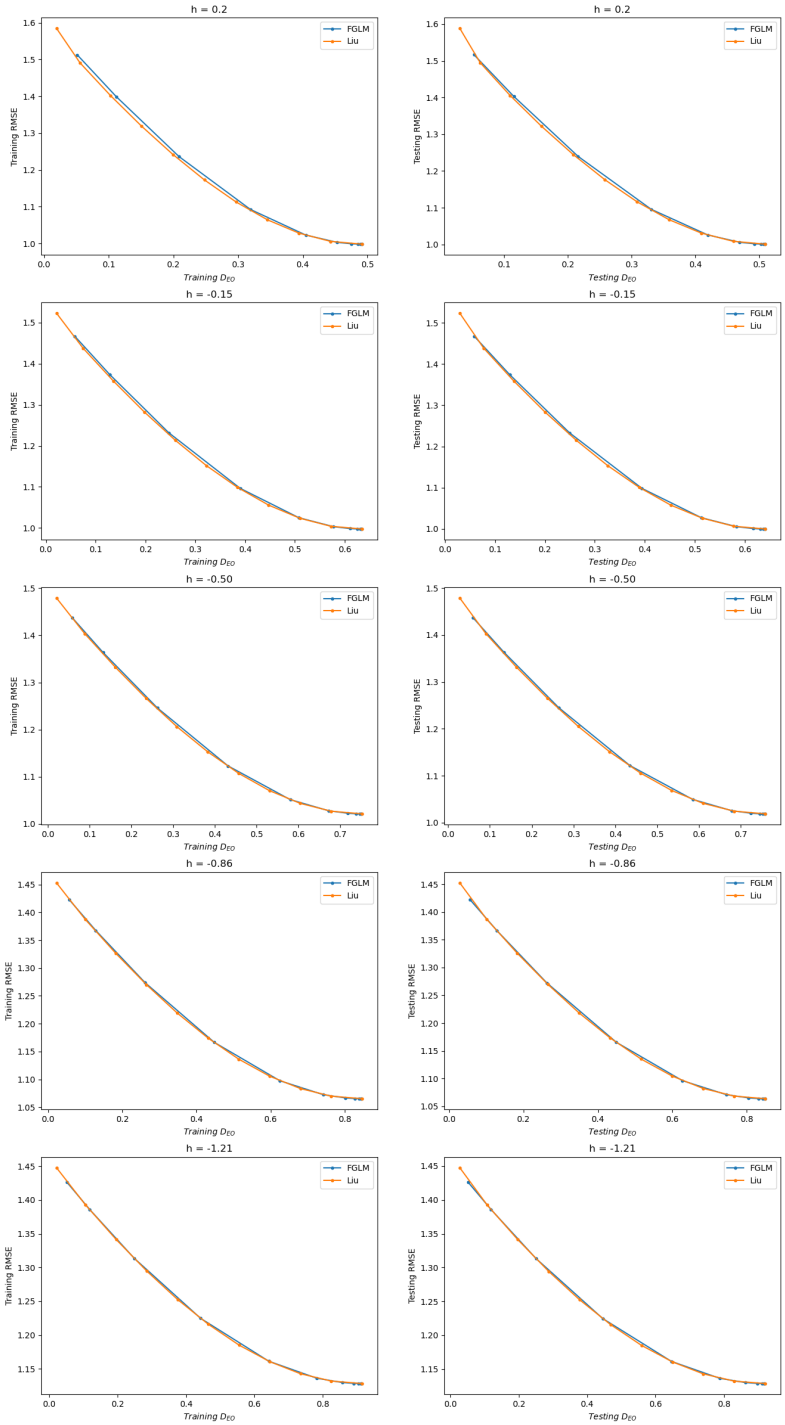

Figure 2.3: Experimental results on the DAG simulated dataset generated with different h values for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{EO}$) on training (left) and testing (right) sets of different linear models fitted without sensitive features. Both values are the lower the better.

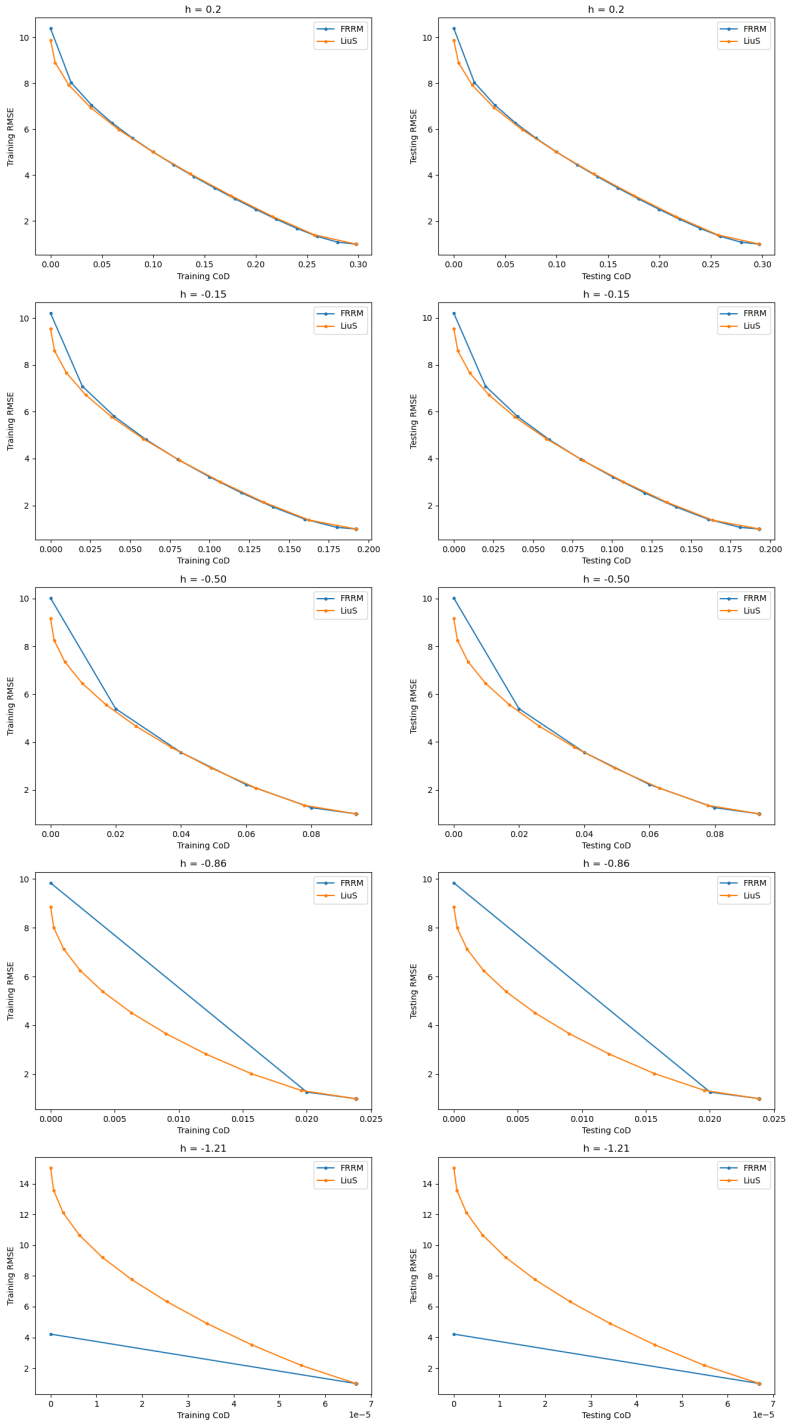

Figure 2.4: Experimental results on the DAG simulated dataset generated with different h values for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by CoD) on training (left) and testing (right) sets of different linear models fitted with sensitive features. Both values are the lower the better.

In the above figures, each curve starts from the bottom-right point, where we have the minimum RMSE and the largest $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ or CoD, respectively. This is the point where the model is equivalent to an ordinary linear regression model. By adjusting the parameters, λ for FGLM, r for FRRM and d for fLiu and fLiuS (fixing $k = 1$), we could achieve a lower measure of fairness while sacrificing the accuracy of the model. Therefore a curve that is more close to the bottom-left of the graph represents a better trade-off between fairness and accuracy in the model.

It can be concluded from Figures 2.2 and 2.3 that when the sensitive feature is excluded from the model, our method fLiu performs similarly as FGLM on this dataset. In Figure 2.2 our curve for fLiu is close to that of FGLM when requirement for fairness (measured by $\mathscr{D}_{LC}$) is not so strict. However, when fairness is low, we cannot beat the accuracy of FGLM in terms of $\mathscr{D}_{LC}$. If measured by $\mathscr{D}_{EO}$, as in Figure 2.3, the fairness can achieved by fLiu is slightly better than FGLM. This pattern is more significant when h increases from -1.21 to 0.2 (when outcomes of the two sensitive groups have larger difference so achieving fairness becomes more important).

On the other hand, look at Figure 2.4, with sensitive features present as inputs in the model, our fLiuS performs similarly as FRRM with large CoD. However, fLiuS shows the potential to work better when CoD is limited to be extremely small. It is also worth mentioning that as h goes from 0.2 to -1.21, the model tends to be more fair with the lowest RMSE. This is not surprising as when $\mathbb{E}(Y|Z = 1) = \mathbb{E}(Y|Z = 0)$ it is much easier to obtain a fair model according to our definitions of fairness. In Figure 2.4 when $h = -1.21$ the ordinary linear regression model itself has a CoD value extremely close to 0 and therefore it is no longer necessary to use FRRM or fLiu anymore.

We include more experimental results in Appendix A.

2.4.2 Experiments on Real Dataset

2.4.2.1 Real Data

Next we try some experiments on the real dataset. The experiment is performed on the law school admission council (lsac) dataset, where we have grade point average

(GPA) of 20,715 students as the continuous outcomes. Only one sensitive feature is selected since in FGLM the fairness is only defined for such settings. Although FRRM and fLiuS can work with multi-dimensional $\mathbf{S}$. The original dataset has a feature race which includes Asian, black, Hispanic, white and other. However, we notice that the most difference of the outcomes (unfairness) in this dataset comes from the black group versus the others, while all non-black groups remain relatively similarly distributed. Therefore we choose to use a binary sensitive feature black versus non-black race in the following experiment. Experiments are also done with all the five races as the sensitive feature and the results are given in Appendix A.

2.4.2.2 Experimental Settings

The data is split into training (70%) and testing (30%) sets. Observations with missing entries have been removed and all continuous predictors are standardised.

In FGLM sensitive features $\mathbf{S}$ are not included as factors to build the prediction model, but only used when calculating the penalty term. Similarly in fLiu we only include non-sensitive features $\mathbf{X}$ in the regression model and sensitive features $\mathbf{S}$ are only used for calculating the optimal parameter d. In FRRM and fLiuS both sensitive features $\mathbf{S}$ and non-sensitive $\hat{\mathbf{U}}$ (the residuals after removing the association between $\mathbf{X}$ and $\mathbf{S}$) are used in the regression model. A linear regression model (LM) with non-sensitive features can be seen as a baseline.

Different than our previous test, this time we try different values of the shrinkage parameter k since in experiments it greatly increases the potential that fairness can be achieved by fLiu/fLiuS when it goes larger than 1. For each fixed k, we can alter the choices of d between default value and d_{fair} that minimises the fairness term and plot a trade-off curve.

We repeat the random train-test splits 20 times. In each split, we build the above models by alternating their parameters. We use RSME to measure the prediction accuracy and we use $\mathscr{D}_{LC}$ and $\mathscr{D}_{EO}$ or CoD to measure the fairness, for scenarios where sensitive features $\mathbf{S}$ are or are not included into the models, respectively.

2.4.2.3 Experimental Results

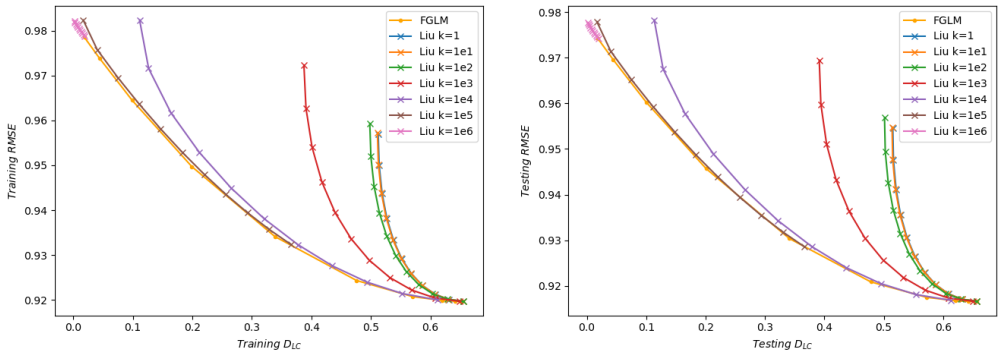

Figure 2.5: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$) on training (left) and testing (right) sets of the lsac dataset with black versus non-black race as sensitive feature of FGLM and fLiu.

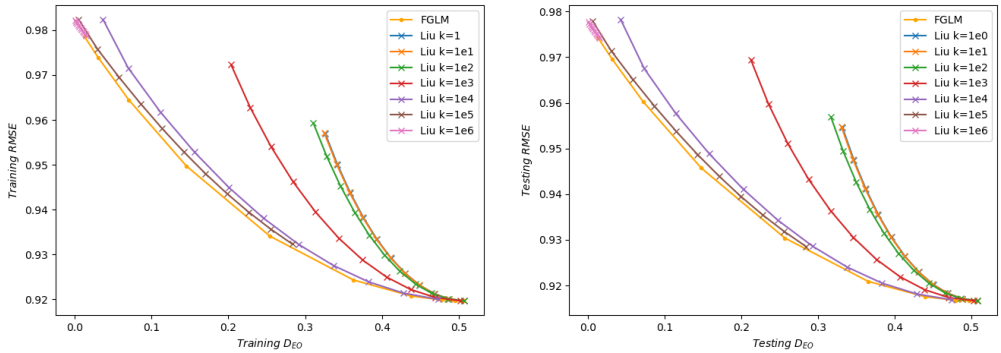

Figure 2.6: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{EO}$) on training (left) and testing (right) sets of the lsac dataset with black versus non-black race as sensitive feature of FGLM and fLiu.

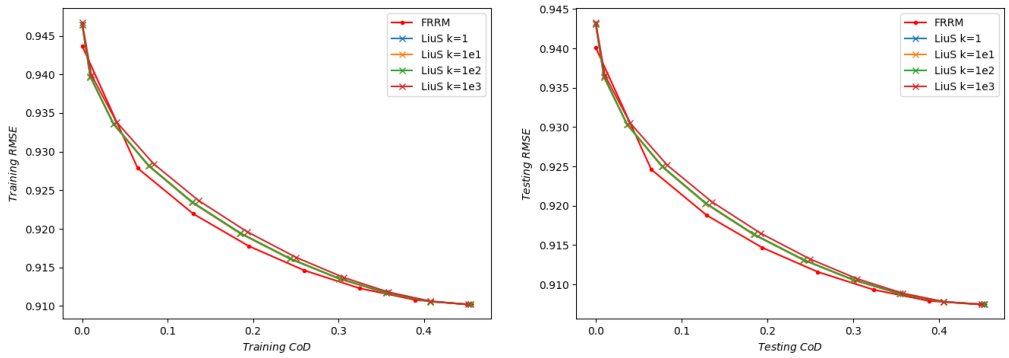

Figure 2.7: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by CoD) on training (left) and testing (right) sets of the lsac dataset with black versus non-black race as sensitive feature of FRRM and fLiuS.

Figures 2.5 , 2.6 and 2.7 show the trade-off curves of different models based on the average results. For fLiu and fLiuS, each line represents one fixed value of k. On each line different points represent different choices of d for fLiu and fLiuS, of which the left most point stands for the value of d that optimise the fairness term with respect to this certian value of k. (We further tried cases where $k < 1$ but the results are very similar to k = 1 and it is hard to tell the differences in the plot, so we did not include them.)

According to Figures 2.5 and 2.6, the ordinary linear regression model (the bottom-right point in each plot) is the most unfair model that minimises RMSE in training dataset (without using sensitive features). Starting from this point, both FGLM and fLiu can achieve better fairness by alternating the parameters, with a loss of MRSE in return. In Figures 2.5, we see that from the most unfair model ($k = 1$), by increasing the value of k (different curves) fLiu can achieve much better fairness with the sacrifice of accuracy and the trade-off is almost identical to that of FGLM. When the value of k is large enough, optimising d can be effective. Overall, by alternating the choice of k and d simultaneously, the shape of the lower bound of the trade-off lines of fLiu can be very close to FGLM in this lsac dataset. This means that fLiu can achieve better fairness without losing too much accuracy compared with FGLM. Looking at Figures 2.6, alternating d gives a better trade-off (more gentle curves) when fairness is measured by $\mathscr{D}_{EO}$. However, FGLM still

slightly outperforms fLiu. This is true since the formulation of FGLM aims to seek an optimal solution to the problem in the whole set of linear models, while fLiu only allow linear models of a specific structure (Liu-type) but is simple and efficient.

Looking at Figures 2.7, the comparison of trade-offs between FRRM and fLiuS shows a similar pattern as in Figures 2.4. fLiuS tends to perform better when d is close to its optimal value d_{fair}. This is when we require the model to be extremely fair with a very small CoD. In addition, increasing k here do help getting better accuracy slightly, but d plays a more important part. In practice, we would suggest to tune k and d according to specific tasks in order to get a better fairness-accuracy trade-off. We also test the results on three additional real datasets, which are included in Appendix A.

Furthermore, we include the average running time for each model to fit on six different datasets in Table 2.3. LSAC, Parkinsons_updrs, Student_performance and Crime are real datasets with continuous outcomes which have been widely considered in fairness learning literature, while DAG and DAG_large stands for our simulated data with different size ($n = 1000$ and $n = 50000$, respectively).

Dataset	lsac	parkinsons_updrs	student_performance	crime	DAG	DAG_large
LM	**0.001704**	**0.003454**	**0.008803**	**0.006281**	**~0**	0.011659
FGLM	0.390665	0.192511	0.041967	0.150179	0.032691	0.398992
FRRM	0.052411	0.021991	0.017122	0.037955	0.022384	0.172484
fLiu	0.003049	0.009538	0.010021	0.013364	0.011373	**0.010978**
fLiuS	0.013547	0.014005	0.012423	0.021840	0.013378	0.035090

Table 2.3: Average computational time of LM, FGLM, FRRM, fLiu and fLiuS on different datasets (in second). (Lowest computational time in each column is marked in bold.)

It can be seen that apart from the linear regression model, our fLiu and fLiuS are the most efficient models followed by FRRM, with FGLM to be the most time consuming one.

2.4.3 Tests of Stability

We have shown that fLiu and fLiuS models benefit from great simplicity and flexibility. We developed the following experiment to show the robustness of the models. We select and fix the training dataset from the lsac data. We fit FRRM and

fLiuS models. In FRRM we use different values of r to bound CoD. In fLiuS we choose different values of k and for each k we calculate and use the optimal value of d that minimises CoD. For ease of notations we record the estimated coefficient parameter vector $\hat{\boldsymbol{\beta}}$ and the predictions $\hat{\mathbf{Y}}$. We then add some noise from the Gaussian distribution with zero mean and standard deviation 0.2 to the numerical non-sensitive standardised features. We fit FRRM and fLiuS again with the noised data and record the coefficient vector and the predictions as $\hat{\boldsymbol{\beta}}'$ and $\hat{\mathbf{Y}}'$. We then calculate the Euclidean distance between $\hat{\boldsymbol{\beta}}'$ and $\hat{\boldsymbol{\beta}}$ and the root of mean squared distance between predictions $\hat{\mathbf{Y}}'$ and $\hat{\mathbf{Y}}$. These are used to measure the stability of the fair model. We expect to have smaller distances between $\hat{\boldsymbol{\beta}}'$ and $\hat{\boldsymbol{\beta}}$ and between $\hat{\mathbf{Y}}'$ and $\hat{\mathbf{Y}}$ in a models that are more robust. The experiment is repeated 100 times and the average results of the above together with the RMSE, CoD and $\mathscr{D}_{LC}$ are reported in Table 2.4.

Model	**Parameter**	**Value**	**Stability**		**Error**	**Fairness**	
			$\|\hat{\boldsymbol{\beta}}' - \hat{\boldsymbol{\beta}}\|$	$\|\hat{\mathbf{Y}}' - \hat{\mathbf{Y}}\|$	RMSE	CoD	$\mathscr{D}_{LC}$
fLiuS	k	1	0.0093	0.0442	0.9541	0.1381	0.5036
		10	0.0092	0.0442	0.9544	0.1317	0.4952
		100	0.0090	0.0437	0.9563	0.0846	0.4324
		1000	0.0077	0.0404	0.9581	0.0052	0.3052
		10000	0.0027	0.0241	0.9594	0.0021	0.1816
		100000	0.0003	0.0050	0.9731	0.0028	0.0383
		1000000	0.0000	0.0006	0.9781	0.0027	0.0043
FRRM	r	0.5	0.0257	0.0478	0.9011	0.5000	0.5479
		0.4	0.0213	0.0477	0.9065	0.4000	0.4890
		0.3	0.0180	0.0476	0.9130	0.3000	0.4403
		0.2	0.0152	0.0476	0.9205	0.2000	0.4002
		0.1	0.0128	0.0475	0.9297	0.1000	0.3683
		0.0	0.0112	0.0475	0.9502	0.0000	0.3506

Table 2.4: Stability, accuracy and fairness measured of fLiuS and FRRM models with different parameters on the lsac dataset with noise of standard deviation 0.2.

We can conclude that with a similar level of fairness (*e.g.* when CoD ≈ 0.1), FRRM gives more accurate predictions while fLiuS gives more stable results with the present of noise. For fLiuS model, larger k (with d_{fair}) leads to better fairness, more stability but worse accuracy.

We also try adjusting the strength of noise with different standard deviations.

Below we show experimental results for the same tests with noises of standard deviations 1 and 0.02, respectively.

Model	Parameter	Value	Stability		Error	Fairness	
			$\|\hat{\boldsymbol{\beta}}' - \hat{\boldsymbol{\beta}}\|$	$\|\hat{\mathbf{Y}}' - \hat{\mathbf{Y}}\|$	RMSE	CoD	$\mathscr{D}_{LC}$
fLiuS	k	1	0.1123	0.1577	0.9676	0.2245	0.4487
		10	0.1121	0.1575	0.9678	0.2149	0.4394
		100	0.1102	0.1559	0.9696	0.1420	0.3698
		1000	0.0974	0.1446	0.9708	0.0087	0.2343
		10000	0.0408	0.0914	0.9677	0.0023	0.1691
		100000	0.0020	0.0243	0.9735	0.0015	0.0501
		1000000	0.0003	0.0030	0.9780	0.0012	0.0063
FRRM	r	0.5	0.3457	0.1843	0.9242	0.5000	0.3969
		0.4	0.2850	0.1792	0.9300	0.4000	0.3507
		0.3	0.2361	0.1755	0.9358	0.3000	0.3139
		0.2	0.1937	0.1728	0.9419	0.2000	0.2847
		0.1	0.1563	0.1710	0.9487	0.1000	0.2615
		0.0	0.1325	0.1693	0.9638	0.0000	0.2473

Table 2.5: Stability, accuracy and fairness measured of fLiuS and FRRM models with different parameters on the lsac dataset with noise of standard deviation 1.

Model	Parameter	Value	Stability		Error	Fairness	
			$\|\hat{\boldsymbol{\beta}}' - \hat{\boldsymbol{\beta}}\|$	$\|\hat{\mathbf{Y}}' - \hat{\mathbf{Y}}\|$	RMSE	CoD	$\mathscr{D}_{LC}$
fLiuS	k	1	0.0003372	0.0045	0.9530	0.1339	0.5078
		10	0.0003364	0.0045	0.9533	0.1277	0.4995
		100	0.0003302	0.0045	0.9552	0.0820	0.4371
		1000	0.0002941	0.0041	0.9570	0.0051	0.3099
		10000	0.0001577	0.0024	0.9588	0.0021	0.1817
		100000	0.0000309	0.0005	0.9731	0.0029	0.0375
		1000000	0.0000035	0.0001	0.9781	0.0029	0.0042
FRRM	r	0.5	0.0007502	0.0048	0.8994	0.5000	0.5587
		0.4	0.0006499	0.0048	0.9047	0.4000	0.4991
		0.3	0.0005754	0.0048	0.9111	0.3000	0.4497
		0.2	0.0005138	0.0048	0.9187	0.2000	0.4090
		0.1	0.0004601	0.0048	0.9280	0.1000	0.3764
		0.0	0.0004238	0.0048	0.9490	0.0000	0.3585

Table 2.6: Stability, accuracy and fairness measured of fLiuS and FRRM models with different parameters on the lsac dataset with noise of standard deviation 0.02.

According to Tables 2.5 and 2.6, we could draw a similar conclusion that given a similar level of fairness measure, although slightly inaccurate in making predictions, in practice, fLiuS are more resilient to noise generated in real data than FRRM. We expect similar results in fLiu comparing with FGLM.

2.4.4 Cross-Validation

Finally, in the following experiment settings we want to try using cross-validation to tune the parameters for fLiu/fLiuS. The idea of adding noise is similar to that in previous work, and the cross-validation process is descried as follows. After splitting the data into train and test sets, we use 5-fold cross-validation to find the k value that gives the best RMSE result in a ridge regression model ($d = 0$). Then with this k value, we alter the value of d to balance accuracy and fairness in our fLiu and fLiuS models. d_{opt} stands for a d value estimated to minimise RMSE suggested by [40] and d_{fair} stands for d value that minimises $\mathscr{D}_{LC}$ or CoD in fLiu and fLiuS, repectively. We further define $d_c = cd_{fair} + (1-c)d_{opt}$ $(0 \leq c \leq 1)$ so that c controls the trade-off between fairness and accuracy through d, with $d_0 = d_{opt}$ and $d_1 = d_{fair}$. In our test with the lsac dateset, the k value after cross-validation would mostly be larger than 1 and smaller than 100. The results are reported in Table 2.7 and Table 2.8 for training and testing datasets, respectively.

Model	Stability		Error	Fairness		
	$\|\hat{\boldsymbol{\beta}}' - \hat{\boldsymbol{\beta}}\|$	$\|\hat{\mathbf{Y}}' - \hat{\mathbf{Y}}\|$	RMSE	CoD	$\mathscr{D}_{LC}$	$\mathscr{D}_{EO}$
LM	0.0118	0.0624	0.9218	-	0.5332	0.3319
fLiu d_{opt}	0.0118	0.0623	0.9218	-	0.5321	0.3312
fLiu $d_{0.2}$	0.0105	0.0586	0.9229	-	0.4993	0.3111
fLiu $d_{0.4}$	0.0113	0.0551	0.9259	-	0.4723	0.2926
fLiu $d_{0.6}$	0.0140	0.0516	0.9310	-	0.4520	0.2760
fLiu $d_{0.8}$	0.0178	0.0482	0.9379	-	0.4393	0.2616
fLiu d_{fair}	0.0220	0.0450	0.9468	-	0.4350	0.2500
fLiuS d_{opt}	0.0109	0.0476	0.9006	0.6335	0.6518	0.5694
fLiuS $d_{0.2}$	0.0105	0.0470	0.9030	0.5420	0.5810	0.4899
fLiuS $d_{0.4}$	0.0101	0.0463	0.9101	0.4250	0.5232	0.4233
fLiuS $d_{0.6}$	0.0097	0.0457	0.9217	0.2894	0.4830	0.3762
fLiuS $d_{0.8}$	0.0093	0.0450	0.9377	0.1663	0.4650	0.3562
fLiuS d_{fair}	0.0091	0.0444	0.9577	0.1132	0.4717	0.3680
FRRM $r = 1.00$	0.0109	0.0476	0.9006	0.6356	0.6539	0.5717
FRRM $r = 0.80$	0.0109	0.0476	0.9006	0.6356	0.6539	0.5717
FRRM $r = 0.60$	0.0321	0.0481	0.9010	0.6000	0.6242	0.5375
FRRM $r = 0.40$	0.0208	0.0478	0.9096	0.4000	0.4896	0.3727
FRRM $r = 0.25$	0.0162	0.0477	0.9195	0.2500	0.4178	0.2717
FRRM $r = 0.18$	0.0144	0.0477	0.9250	0.1800	0.3911	0.2285
FRRM $r = 0.13$	0.0132	0.0477	0.9294	0.1300	0.3746	0.1990
FRRM $r = 0.08$	0.0121	0.0477	0.9346	0.0800	0.3606	0.1711
FRRM $r = 0.04$	0.0114	0.0477	0.9398	0.0400	0.3518	0.1515
FRRM $r = 0.00$	0.0109	0.0476	0.9533	0.0000	0.3489	0.1446

Table 2.7: Stability, accuracy and fairness measured of LM, fLiu, fLiuS (with cross-validation k) and FRRM models with different parameters on the lsac training dataset with noise of standard deviation 0.2.

Model	Stability		Error	Fairness		
	$\|\hat{\boldsymbol{\beta}}' - \hat{\boldsymbol{\beta}}\|$	$\|\hat{\mathbf{Y}}' - \hat{\mathbf{Y}}\|$	RMSE	CoD	$\mathscr{D}_{LC}$	$\mathscr{D}_{EO}$
LM	0.0118	0.0624	0.9187	-	0.5331	0.3262
fLiu d_{opt}	0.0118	0.0623	0.9187	-	0.5320	0.3255
fLiu $d_{0.2}$	0.0105	0.0587	0.9198	-	0.5003	0.3069
fLiu $d_{0.4}$	0.0113	0.0551	0.9229	-	0.4745	0.2900
fLiu $d_{0.6}$	0.0140	0.0516	0.9280	-	0.4557	0.2752
fLiu $d_{0.8}$	0.0178	0.0482	0.9351	-	0.4447	0.2630
fLiu d_{fair}	0.0220	0.0450	0.9440	-	0.4421	0.2535
fLiuS d_{opt}	0.0109	0.0477	0.8974	0.6311	0.6516	0.5678
fLiuS $d_{0.2}$	0.0105	0.0470	0.8999	0.5397	0.5811	0.4884
fLiuS $d_{0.4}$	0.0101	0.0464	0.9070	0.4228	0.5236	0.4218
fLiuS $d_{0.6}$	0.0097	0.0457	0.9186	0.2873	0.4837	0.3746
fLiuS $d_{0.8}$	0.0093	0.0451	0.9345	0.1638	0.4659	0.3546
fLiuS d_{fair}	0.0091	0.0444	0.9546	0.1096	0.4727	0.3664
FRRM $r = 1.00$	0.0109	0.0477	0.8974	0.6332	0.6537	0.5701
FRRM $r = 0.80$	0.0109	0.0477	0.8974	0.6332	0.6537	0.5701
FRRM $r = 0.60$	0.0321	0.0481	0.8978	0.5975	0.6241	0.5359
FRRM $r = 0.40$	0.0208	0.0479	0.9063	0.3980	0.4899	0.3710
FRRM $r = 0.25$	0.0162	0.0478	0.9162	0.2487	0.4188	0.2700
FRRM $r = 0.18$	0.0144	0.0477	0.9217	0.1790	0.3925	0.2268
FRRM $r = 0.13$	0.0132	0.0477	0.9261	0.1293	0.3763	0.1973
FRRM $r = 0.08$	0.0121	0.0477	0.9313	0.0796	0.3626	0.1695
FRRM $r = 0.04$	0.0114	0.0477	0.9365	0.0398	0.3540	0.1500
FRRM $r = 0.00$	0.0109	0.0477	0.9497	0.0000	0.3514	0.1438

Table 2.8: Stability, accuracy and fairness measured of LM, fLiu, fLiuS (with cross-validation k) and FRRM models with different parameters on the lsac testing dataset with noise of standard deviation 0.2.

According to Tables 2.7 and 2.8, there is no significant difference between the performance of any method in the training and testing sets. We first look at the three criteria of fairness, CoD, $\mathscr{D}_{LC}$ and $\mathscr{D}_{EO}$. According to Table 2.8, when c and r change from 1 to 0 (for fLiu/fLiuS and FRRM, respectively), all three fairness measures decrease. Therefore all models can achieve fairness effectively. However, if one model has a lower fairness than another model in one fairness measure, it is not necessarily more fair when measured in other criteria. For example, FRRM with $r = 0.18$ results in 0.18 in CoD, around 0.39 in $\mathscr{D}_{LC}$ and 0.23 in $\mathscr{D}_{EO}$. However, given a slightly smaller CoD $= 0.16$, the fLiuS model with $c = 0.8$ has larger $\mathscr{D}_{LC}$ and $\mathscr{D}_{EO}$, which are 0.47 and 0.35, respectively. In contrast, with a similar level of $\mathscr{D}_{LC} = 0.48$, for example, fLiuS model with $c = 0.6$ gives 0.29 in CoD while FRRM with $r = 0.40$ has CoD $= 0.40$. Second, comparing fLiuS and fLiu with $d = d_{opt}$, we conclude that adding the sensitive feature to the model helps achieve a lower RMSE for this dataset. If we change the parameters to get better fairness, *e.g.* with $\mathscr{D}_{LC} \approx 0.48$, both fLiuS and FRRM have lower RMSE than fLiu does. This suggests that including sensitive features can improve model fairness with less influence on prediction accuracy. However, more evidence is needed for this conclusion. Finally, with a similar level of fairness (under any fairness measure), fLiu and fLiuS models give a higher RMSE but better stability than FRRM model. These conclusions are similar to what we have in Tables 2.4 to 2.6. Note that here we only give an example of how the parameters can be tuned using cross-validation in our models. Further work can be done to explore other methods for choosing the parameters or to give mathematical support to them.

2.5 Conclusions

In this chapter, we discussed fair linear regression models. We first introduced fairness in regression by giving some definitions, followed by a summary of two state-of-the-art methods, FGLM and FRRM. We then proposed two models based on Liu-type estimator, fLiu and fLiuS, intended for modelling with or without sensitive features, respectively. Experimental results were given to show the trade-off

between accuracy and fairness compared with other methods. We conclude that for models without using sensitive features as input, the performance of fLiu can be close to FGLM in some dataset, while enjoys great simplicity. For models including sensitive features as input, fLiuS can outperform FRRM in simple datasets, especially with strict fairness restrictions. What is more, the simplicity of fLiu and fLiuS means that they can work efficiently and experiments show that they are more resilient to noise and therefore more stable. Future work can be done to develop a method to tune the values of the hyper-parameters k and d according to specific applications. It is also interesting to consider a metric used to evaluate the trade-off between accuracy and fairness by linking these two terms together.

Chapter 3

Fair Generalised Linear Model with the Maximum Mean Discrepancy Penalty

3.1 Introduction

Generalised linear model (GLM) is a powerful tool in statistical modelling and machine learning. It offers a flexible framework for modelling the relationship between the response variable and the predictors by generalising the ordinary linear model in two aspects. Firstly, the response variable follows some distribution from the exponential family, which can be chosen to provide an appropriate variance structure. Secondly, the conditional mean of the distribution is linked with the linear predictor via a link function. Compared with ordinary linear regression, these extensions make the model more flexible and accommodate a variety of different data types, including continuous (Gaussian distribution), binary (Bernoulli distribution), categorical and count (Poisson distribution) outcomes. The generalised linear model can be efficiently fitted by using a Newton's method to minimise the negative log-likelihood. Due to its simplicity and flexibility, the generalised linear model has been widely used for decision making in various fields such as finance, medicine, and social sciences.

However, a model can be problematic if it gives different predictions for dif-

ferent demographic groups, leading to a critical concern about fairness. For example, this can lead to unfair decisions in lending, hiring, and medical diagnosis for individuals based on different races, genders, or religions, which are considered sensitive attributes. Despite the flexibility and accuracy in making predictions, standard generalised linear models, like many other models, often fail to address such concerns.

Fairness can be defined in various ways according to specific application scenario. Studies have been done to explore different measures of fairness over the past decades, and several learning methods have been proposed to achieve these goals. One notable approach is the fair generalised linear model with a convex penalty (FGLM) introduced by [20]. By incorporating a convex fairness penalty within the generalised linear model framework, it ensures that the predictions remain fair between different protected groups while maintaining good predictive accuracy.

FGLM is inspired based on two fairness criteria, namely equalised expected outcomes ($\mathscr{D}_{EO}$) and equalised expected log-likelihoods ($\mathscr{D}_{ELL}$), neither of which can be optimised efficiently as a fairness penalty due to non-convexity. [20] uses a different penalty term, equalised linear components ($\mathscr{D}_{LC}$), in their FGLM framework, arguing that this formulation leads to efficient computation while efficiently optimising the bounds of both $\mathscr{D}_{EO}$ and $\mathscr{D}_{ELL}$. However, the penalty term in FGLM can not fully capture complex distributional differences between different sensitive groups, because it only captures the first two moments of the distributions of predicted outcomes. As a result, these methods can struggle in some settings, especially when sensitive attributes influence the data in a non-linear fashion.

To address these limitations, we propose a new fairness-aware GLM framework that uses maximum mean discrepancy (MMD) as the fairness penalty. MMD is a powerful kernel-based statistical distance metric that measures the difference between probability distributions in a reproducing kernel Hilbert space (RKHS) [47]. By incorporating MMD into the optimisation process of GLMs, we ensure that the predicted distributions for different sensitive groups are close to each other, while maintaining convexity of the loss function so that the optimisation remains

computationally efficient. Unlike the fairness penalty in FGLM, MMD can capture higher-order distributional differences, and is thus more robust and powerful for fair learning. Additionally, our framework is also based on GLMs and thus can be applied to both classification and regression tasks, making it suitable for a wide range of applications.

The remainder of this chapter is structured as follows. Section 3.2 provides a summary of related works in this field. In Section 3.3, we first introduce the notations we use throughout this chapter, followed by a introduction of the fair generalised linear model algorithm by [20]. We then introduce the fairness measure we propose to use, namely the maximum mean discrepancy (MMD). Our proposed MMD-regularised GLM is then introduced, with descriptions in its mathematical formulation and optimisation method. Section 3.4 presents experimental results on real datasets with different outcome types for various tasks, demonstrating the effectiveness of our approach compared with FGLM. Finally, Section 3.5 concludes this chapter with discussions on potential future research directions.

3.2 Related Work

A number of researches have been done to address the fairness problem in machine learning over the past few years. These methods can be arranged into three categories according to the time where fairness is enhanced: pre-process, in-process, and post-process [10, 43]. We will focus on the cases where fairness is considered during the learning process. This is usually achieved by modifying the loss function to penalise unfair predictions.

Many early works deal with binary classification problems. For example, [48] proposed to add regularisers to reduce mutual information between predictions and the sensitive feature. [14] and [15] used constraints to avoid disparate mistreatment and disparate impact. [11] considered demographic parity and equalized odds and reduced the fair binary classification problem into a sequence of unconstrained cost-sensitive classification problems.

Attempts have also been made to tackle regression tasks. [17] introduced a

flexible family of penalties including group fairness, individual fairness, and a hybrid notion of both and applied them to regression problems. [16] used statistical parity and bounded group loss as fairness notions and extended the work of [11] to regression tasks. [19] proposed a projection-based fair learning approach using the Hilbert Schmidt independence criterion as fairness, which allows the method to work with non-linear regression.

[20] proposed to achieve fairness in GLMs by including a convex penalty term in the loss function. It was the first work that is based on GLMs, which give it the flexibility to be able to handle a broad range of outcome types. They adapted the definition of fairness by adding a variance term to the expected difference of the pairwise linear components. In this way, the loss function remains convex so that a solution can be found efficiently. According to [20], both fairness criteria, equalised expected outcomes and equalised expected log-likelihoods, are bounded by their formulation and can be effectively reduced by changing the weighting parameter. However, such definition of the penalty only takes into account the first and second moments of the distributions of predicted outcomes from different sensitive groups. We seek a more robust fairness measure that works well with data from all kinds of distributions.

Maximum mean discrepancy (MMD) is a kernel-based statistical measure that provides an alternative way to determine whether two samples are from the same or different distributions [47]. It is a non-parametric method and does not introduce any prior so is flexible in dealing with datasets coming from a wider range. It is defined as the maximum difference in expectations over the unit balls in a characteristic reproducing kernel Hilbert space (RKHS). This definition means that it can capture more than mean differences and variance differences as traditional fairness measures do. For example, [49] presented a fair learning algorithm based on MMD constraints and showed that their method outperforms other state-of-the-art techniques. Therefore, we propose to use MMD as an alternative fairness penalty term in our GLM framework. Unlike [49], which uses MMD to align the distributions of feature representations across different sensitive groups, we use MMD

to monitor distributional differences in the predicted outcomes. The loss function constructed in this way is convex with respect to the coefficient vector of the GLM so the optimisation can be done efficiently.

3.3 Methodology

3.3.1 Fair Generalised Linear Model with a Convex Penalty

In this section, we introduce the notations we use for this chapter and briefly summarise the fair generalised linear model proposed by [20].

Let $\mathbf{X} \in \mathbb{R}^{N\times d}$ be predictor variables with number of observations N and number of non-sensitive features d. Let $\mathbf{Y} \in \mathbb{R}^{N\times 1}$ be the vector of outcomes. We write $\mathscr{Y}$ for the set of all possible outcomes. Note that $\mathscr{Y}$ is discretised into small segments for continuous response. Denote the set of all possible values of the sensitive feature by $\mathscr{A} = a_1, \ldots, a_K$ where K is the number of classes of the sensitive feature. We use $\mathbf{X}^k$ and $\mathbf{Y}^k$ to represent the predictor and response variables of sensitive group a_k, respectively, and $\mathbf{X}^{ky}$ are the predictor variables of sensitive group a_k which have label y. We write $\mathbf{x}_i$ and y_i for the ith instance, $1 \leq i \leq N$. Let $\boldsymbol{\beta} \in \mathbb{R}^d$ be the coefficient vector and μ be the expected value of the response variable in a generalised linear model.

[20] proposed their fair generalised linear model where fairness is achieved by adding a convex penalty term, $\mathscr{D}_{LC}$, measuring the average expectation of squared difference of the linear predictors in two sensitive groups with the same outcome,

$$\mathscr{D}_{LC} = \frac{1}{\kappa} \sum_{\substack{k,l=1 \\ k<l}}^{K} \sum_{y\in\mathscr{Y}} \mathbb{E}((\mathbf{X}^{ky}\boldsymbol{\beta} - \mathbf{X}^{ly}\boldsymbol{\beta})^2),$$

where $\kappa = \frac{K(K-1)}{2}|\mathscr{Y}|$ is the total number of possible combinations where we pick two sensitive groups a_i and a_j together with one outcome y. The objective function

to minimise is given by

$$\begin{aligned} L_{\text{FGLM}} &= -\mathbb{E}(\ell(\boldsymbol{\beta};\mathbf{X},\mathbf{Y})) + \lambda \cdot \mathscr{D}_{LC} \\ &= -\mathbb{E}(\ell(\boldsymbol{\beta};\mathbf{X},\mathbf{Y})) + \frac{\lambda}{\kappa} \sum_{\substack{k,l=1 \\ k<l}}^{K} \sum_{y\in\mathscr{Y}} \mathbb{E}((\mathbf{X}^{ky}\boldsymbol{\beta} - \mathbf{X}^{ly}\boldsymbol{\beta})^2) \\ &\approx -\frac{1}{N}\sum_{i=1}^{n} \ell(\boldsymbol{\beta};\mathbf{x}_i,y_i) + \frac{\lambda}{\kappa} \sum_{\substack{k,l=1 \\ k<l}}^{K} \sum_{y\in\mathscr{Y}} \frac{1}{n^{kly}} \sum_{(i,j)\in S^{kly}} (\mathbf{x}_i\boldsymbol{\beta} - \mathbf{x}_j\boldsymbol{\beta})^2, \end{aligned} \tag{3.1}$$

where S^{kly} is the set of pairs (i,j) such that $y_i = y_j = y$ and $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to sensitive groups a_k and a_l, respectively, and n_{kly} is the number of pairs in S^{kly}. $\ell(\boldsymbol{\beta};\mathbf{x}_i,y_i)$ is the log-likelihood as defined in GLM. Finally, λ is a positive weighting parameter that we choose to balance the trade-off between predictive accuracy and fairness.

[20] solved the optimisation problem of minimising L_{FGLM} in (3.1) with a Newton-Raphson method. The algorithm works efficiently and convergence is guaranteed as the function is convex in $\boldsymbol{\beta}$.

Each element of the penalty term $\mathscr{D}_{LC}$ for fairness can be written as follows,

$$\mathbb{E}((\mathbf{X}^{ky}\boldsymbol{\beta} - \mathbf{X}^{ly}\boldsymbol{\beta})^2) = \text{Var}(\mathbf{X}^{ky}\boldsymbol{\beta} - \mathbf{X}^{ly}\boldsymbol{\beta}) + (\mathbb{E}(\mathbf{X}^{ky}\boldsymbol{\beta}) - \mathbb{E}(\mathbf{X}^{ly}\boldsymbol{\beta}))^2,$$

where it consists of two terms, the variance of difference in the linear predictors from different sensitive groups and the squared difference in the expectations of the linear predictors. The squared difference term comes intuitively from the definition of equalised expected outcomes in linear regression and, according to [20], introducing the variance term help achieving a lower difference in the expected log-likelihoods of the model and thus resulting better fairness. However, the definitions of fairness can vary and using the expectation of difference in the linear predictors together with their variance as a measure still fails to capture the whole plot of the resulting model. Therefore, we seek to quantify the fairness of the generalised linear model with a different measure that estimate how close the distributions of $\mu(\mathbf{X}^{ky}\boldsymbol{\beta})$ and $\mu(\mathbf{X}^{ly}\boldsymbol{\beta})$ given label y are, namely the maximum mean discrepancy

(MMD), where μ is the conditional mean of the distribution depending on the predictor variables $\mathbf{X}$ and the choice of the link function.

3.3.2 Maximum Mean Discrepancy

In this section, we introduce the maximum mean discrepancy to measure the distance between two distributions.

Let X and Y be random variables defined on $\mathscr{X}$ with probability measures p and q, respectively. The squared maximum mean discrepancy between p and q is defined as,

$$\text{MMD}^2(p,q) = ||\mu_p - \mu_q||^2_{\mathscr{H}},$$

where $\mathscr{H}$ is an reproducing kernel Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathscr{H}}$ and μ_p and μ_p are mean embeddings of p and q, respectively.

Suppose X and X' are independent and identically distributed random variables from p and Y and Y' are independent and identically distributed random variables from q. The squared population maximum mean discrepancy is given by,

$$\text{MMD}^2(p,q) = \mathbb{E}(\text{K}(X,X')) + \mathbb{E}(\text{K}(Y,Y')) - 2\mathbb{E}(\text{K}(X,Y)),$$

where $\text{K}(\cdot,\cdot) : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is some positive definite kernel (*e.g.* RBF kernel) such that $\text{K}(x,x') = \langle \phi(x), \phi(x') \rangle_{\mathscr{H}}$ and $\phi(\cdot)$ is the corresponding feature map.

An unbiased empirical estimate of the squared maximum mean discrepancy can be obtained by,

$$\widehat{\text{MMD}^2}(x,y) = \frac{1}{n(n-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \text{K}(x_i,x_j) + \frac{1}{m(m-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{m} \text{K}(y_i,y_j) - \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \text{K}(x_i,y_j),$$

where x_i $(1 \leq i \leq n)$ and y_j $(1 \leq j \leq m)$ are independent samples from p and q, respectively, and n and m are number of observations.

As a non-parametric distance measure, maximum mean discrepancy does not make any assumptions about the distributions of the data. Instead of capturing just

mean differences or variance as many traditional metrics do, maximum mean discrepancy can capture higher-order differences, and thus turns to be a more comprehensive fairness measure. It can also be estimated when the sample sizes from both distributions are different. The choice of kernel function provides flexibility, allowing it to capture more complex distributional differences. In addition, maximum mean discrepancy is effective for both continuous and categorical outcomes as well as in high-dimensional settings. Therefore, we choose to use it as a fairness criterion to build our fair generalised linear model.

3.3.3 Fair Generalised Linear Model with the Maximum Mean Discrepancy Penalty

In this section, we present our fair generalised linear model with a penalty based on the maximum mean discrepancy described above.

We define the sample squared maximum mean discrepancy of a generalised linear model between two sensitive groups a_k and a_l with the same label y as follows,

$$\begin{aligned}\widehat{\mathrm{MMD}^2}(\boldsymbol{\beta};k,l,y) = &\frac{1}{n(n-1)} \sum_{\substack{(i,j)\in S^{kky} \\ i\neq j}} \mathrm{K}(\mu(\mathbf{X}_i\boldsymbol{\beta}),\mu(\mathbf{X}_j\boldsymbol{\beta})) \\ &+ \frac{1}{m(m-1)} \sum_{\substack{(i,j)\in S^{lly} \\ i\neq j}} \mathrm{K}(\mu(\mathbf{X}_i\boldsymbol{\beta}),\mu(\mathbf{X}_j\boldsymbol{\beta})) \\ &- \frac{2}{nm} \sum_{(i,j)\in S^{kly}} \mathrm{K}(\mu(\mathbf{X}_i\boldsymbol{\beta}),\mu(\mathbf{X}_j\boldsymbol{\beta}))\end{aligned}$$

where μ is the link function, $\mathrm{K}(\cdot,\cdot)$ is some kernel, and n and m are number of observations of $\mathbf{X}^{ky}$ and $\mathbf{X}^{ly}$, respectively.

Then we define the loss function of our fair generalised linear model as,

$$L_{\mathrm{MMD}} = -\frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{\beta};\mathbf{X}_i,Y_i) + \frac{\lambda}{\kappa} \sum_{\substack{k,l=1 \\ k<l}}^{K} \sum_{y\in\mathscr{Y}} \widehat{\mathrm{MMD}^2}(\boldsymbol{\beta};k,l,y). \tag{3.2}$$

The choice of kernel decides how differences between distributions are mea-

sured and can influence the performance of the model. The radial basis function (RBF) kernel is widely used due to its smoothness and universal approximating ability. It is a characteristic kernel so that it can distinguish two distributions uniquely. It is also able to capture distributional differences in higher orders. Other choices of kernels (*e.g.* polynomial kernel) can be explored for adapting to varying data characteristics, although differentiability is also crucial for efficient optimisation.

We use the RBF kernel, $\mathrm{K}(\mathbf{X}_i\boldsymbol{\beta},\mathbf{X}_j\boldsymbol{\beta}) = \exp(-\frac{(\mathbf{X}_i\boldsymbol{\beta}-\mathbf{X}_j\boldsymbol{\beta})^2}{2\sigma^2})$, in our experimental settings. It has a parameter σ, known as the bandwidth. It controls the scale of the kernel function and can be selected using techniques such as cross-validation. A small bandwidth leads to a more sensitive detection of distributional differences but may also lead to a model that is more sensitive to noise. Assume that the response of the model follows a normal distribution with the identity link function, *i.e.* $\mu(\mathbf{X}_i\boldsymbol{\beta}) = \mathbf{X}_i\boldsymbol{\beta}$ for $1 \leq i \leq N$. The maximum mean discrepancies for different distributions and their derivatives can be obtained in a similar way by replacing the link functions. The gradient,

$$\nabla\widehat{\mathrm{MMD}^2}(\boldsymbol{\beta};k,l,y) = \frac{1}{n(n-1)}\sum_{\substack{(i,j)\in S^{aay}\\ i\neq j}}\nabla_{ij} + \frac{1}{m(m-1)}\sum_{\substack{(i,j)\in S^{bby}\\ i\neq j}}\nabla_{ij} - \frac{2}{nm}\sum_{(i,j)\in S^{aby}}\nabla_{ij},$$

where ∇_{ij} is given by,

$$\begin{aligned}\nabla_{ij} &= \frac{\partial}{\partial\boldsymbol{\beta}}\mathrm{K}(\mathbf{X}_i\boldsymbol{\beta},\mathbf{X}_j\boldsymbol{\beta})\\ &= \exp(-\frac{(\mathbf{X}_i\boldsymbol{\beta}-\mathbf{X}_j\boldsymbol{\beta})^2}{2\sigma^2})\frac{(\mathbf{X}_j\boldsymbol{\beta}-\mathbf{X}_i\boldsymbol{\beta})(\mathbf{X}_i-\mathbf{X}_j)}{\sigma^2}\\ &= \mathrm{K}(\mathbf{X}_i\boldsymbol{\beta},\mathbf{X}_j\boldsymbol{\beta})\underbrace{\frac{-(\mathbf{X}_i-\mathbf{X}_j)\boldsymbol{\beta}(\mathbf{X}_i-\mathbf{X}_j)}{\sigma^2}}_{\mathbf{Q}_{ij}}.\end{aligned}$$

Similarly, we can find the Hessian,

$$\nabla^2_{ij} = \nabla_{ij}\mathbf{Q}^T_{ij} + \mathrm{K}(\mathbf{X}_i\boldsymbol{\beta},\mathbf{X}_j\boldsymbol{\beta})\frac{-(\mathbf{X}_i-\mathbf{X}_j)(\mathbf{X}_i-\mathbf{X}_j)}{\sigma^2}.$$

Therefore, to minimise the loss function in (3.2), we can set $\boldsymbol{\beta}_0 = \mathbf{0}$ and

then iteratively update $\boldsymbol{\beta}$ using the Newton-Raphson method by $\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i - (\nabla^2 L_{\text{MMD}}(\boldsymbol{\beta}_i))^{-1} \nabla L_{\text{MMD}}(\boldsymbol{\beta}_i)$, where

$$\nabla L_{\text{MMD}}(\boldsymbol{\beta}) = -\frac{1}{N}\mathbf{X}^T(\mathbf{Y}-\boldsymbol{\mu}) + \frac{\lambda}{\kappa}\sum_{\substack{k,l=1\\k<l}}^{K}\sum_{y\in\mathscr{Y}} \nabla\widehat{\text{MMD}^2}(\boldsymbol{\beta};k,l,y).$$

and

$$\nabla^2 L_{\text{MMD}}(\boldsymbol{\beta}) = \frac{1}{N}\mathbf{X}^T \text{Diag}(\boldsymbol{\mu})\mathbf{X} + \frac{\lambda}{\kappa}\sum_{\substack{k,l=1\\k<l}}^{K}\sum_{y\in\mathscr{Y}} \nabla^2\widehat{\text{MMD}^2}(\boldsymbol{\beta};k,l,y).$$

Note that $\mathbf{0}$ is the all-zeros vector, $\boldsymbol{\mu}$ is the mean response vector of the generalised linear model for all observations and $\text{Diag}(\boldsymbol{\mu})$ is an $N \times N$ diagonal matrix with the elements of $\boldsymbol{\mu}$ on the main diagonal.

For responses of different types such as binary, count or multi-class, a proper link function should be chosen and the same algorithm applies.

3.4 Experimental Results

In this section, we compare the performance of our fair generalised linear model with the maximum mean discrepancy penalty (FGLM-MMD) with the fair generalised linear model (FGLM) by [20].

We perform our tests on two datasets with different types of outcomes. The first dataset is the Law School Admission Council (lsac) dataset. It consists of 20715 observations of student grade point average (GPA) as targets. Each instance has 7 features including race as a sensitive feature with five classes. The outcomes are continuous, therefore we fit generalised linear models with normal distribution using identity link function $\mu(\mathbf{X}_i\boldsymbol{\beta}) = \mathbf{X}_i\boldsymbol{\beta}$.

The second dataset we use is the German credit dataset with binary outcomes. It contains records of 1000 individuals whose credit risks are classified as either good or bad. The predictors consist of 46 features after one-hot encoding, including age, employment status, credit history, number of people being liable to provide maintenance for, etc. We use sex as the sensitive feature. For this binary classifica-

tion task we use Bernoulli distribution and logit link function $\mu(\mathbf{X}_i\boldsymbol{\beta}) = \frac{1}{1+\exp(\mathbf{X}_i\boldsymbol{\beta})}$. In addition, for both datasets, the sensitive feature is not included in predictors $\mathbf{X}$, but only used to calculate the penalty term to help obtain the fair coefficient vector $\boldsymbol{\beta}$.

We split each dataset into training (70%) and testing (30%) sets. We use the training data to build FGLM and FGLM-MMD with different parameters. Both FGLM and FGLM-MMD have weighting parameter $\lambda \geq 0$ that controls the trade-off between accuracy and fairness. FGLM-MMD has another parameter σ which is defined in the RBF kernel. We use each model we build to predict the outcomes and calculate the training and testing accuracies. There is currently no single evaluation of fairness that has been widely accepted. Therefore, we consider the following measures (estimates),

$$\mathscr{D}_{EO} = \frac{1}{\kappa}\sum_{\substack{k,l=1\\k<l}}^{K}\sum_{y\in\mathscr{Y}}(\mathbb{E}(\mu(\mathbf{X}^{ky}\boldsymbol{\beta})) - \mathbb{E}(\mu(\mathbf{X}^{ly}\boldsymbol{\beta})))^2,$$

$$\mathscr{D}_{LC} = \frac{1}{\kappa}\sum_{\substack{k,l=1\\k<l}}^{K}\sum_{y\in\mathscr{Y}}\mathbb{E}((\mathbf{X}^{ky}\boldsymbol{\beta} - \mathbf{X}^{ly}\boldsymbol{\beta})^2),$$

$$\text{MMD} = \frac{1}{\kappa}\sum_{\substack{k,l=1\\k<l}}^{K}\sum_{y\in\mathscr{Y}}\widehat{\text{MMD}^2}(\boldsymbol{\beta};k,l,y),$$

and for regression task with continuous outcomes,

$$\text{KS} = \frac{1}{\kappa}\sum_{\substack{k,l=1\\k<l}}^{K}\sum_{y\in\mathscr{Y}}\mathbb{D}(\boldsymbol{\beta};k,l,y),$$

where $\mathbb{D}(\boldsymbol{\beta};k,l,y)$ is the two-sample Kolmogorov–Smirnov (KS) statistic between predictions of the model of sensitive groups a_k and a_l with label y. Note that FGLM aims to minimise $\mathscr{D}_{LC}$ and our FGLM-MMD is designed to minimise MMD with a specific σ.

The data split is repeated 20 times and we plot the average accuracy-fairness

trade-offs of FGLM and FGLM-MMD with different σ for lsac and German credit training and testing datasets in Figures 3.1-3.4, respectively. In each figure, accuracy is plotted on the vertical axis and fairness measured by different metrics is shown on the horizontal, both of which are the lower the better. Accuracy for the lsac dataset (Figures 3.1 and 3.2) is measured by RMSE and the German credit dataset (Figures 3.3 and 3.4) is calculated as the predictive error rate.

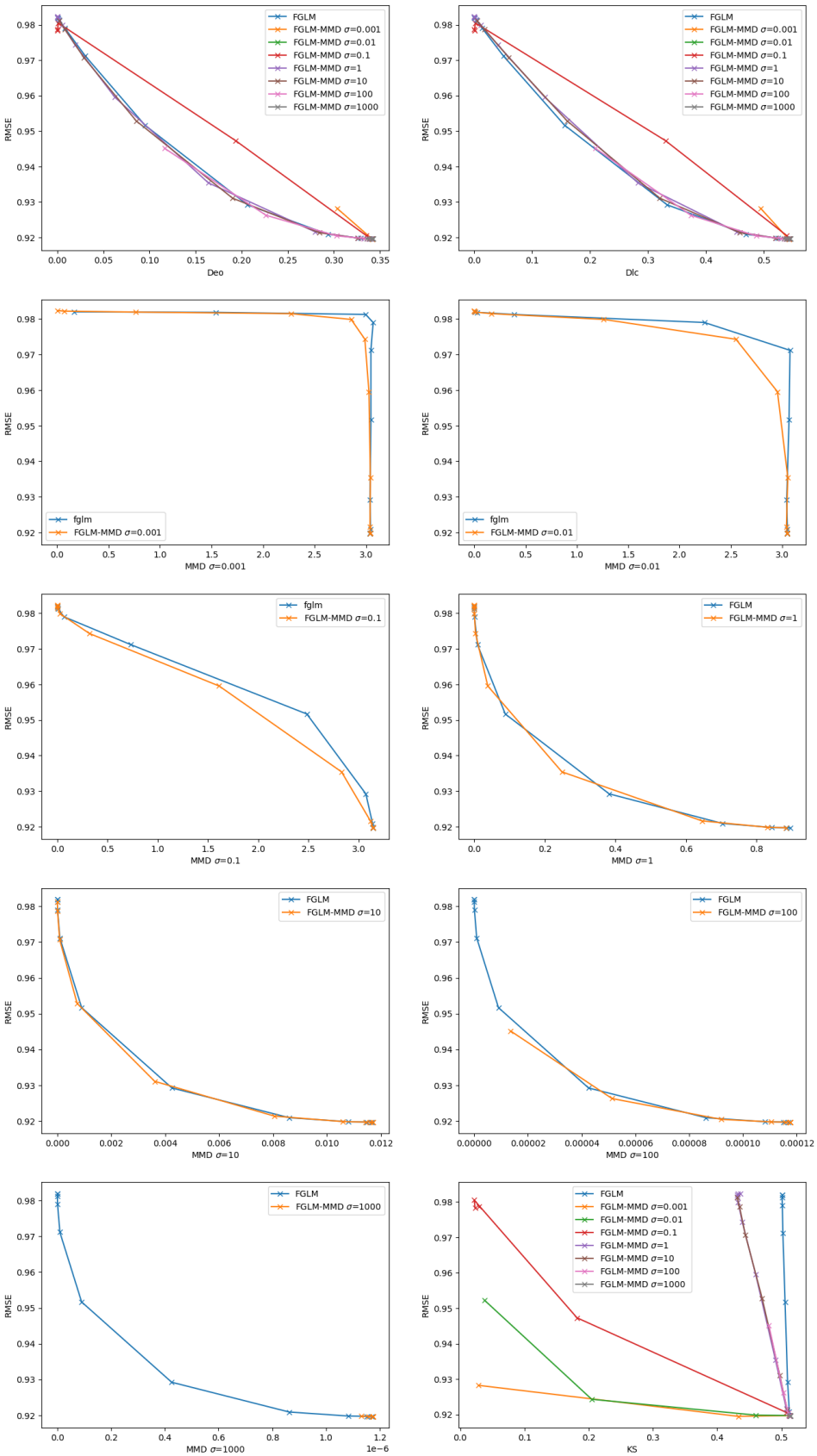

Figure 3.1: Average experimental results for trade-offs of prediction accuracy and fairness on training sets of the lsac dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by RMSE, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Middle rows and bottom left: MMD with different values of σ. Bottom right: KS statistic.

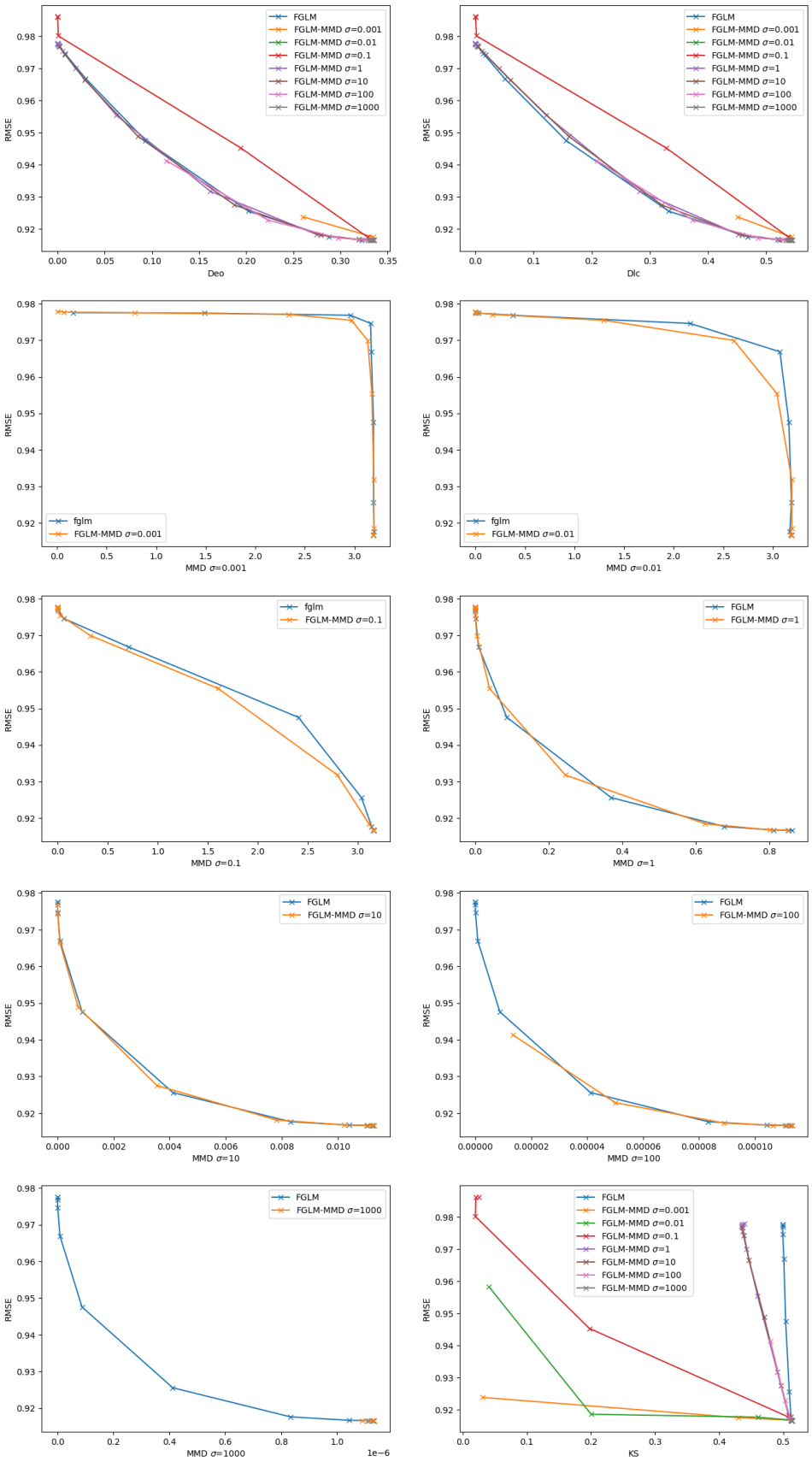

Figure 3.2: Average experimental results for trade-offs of prediction accuracy and fairness on testing sets of the lsac dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by RMSE, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Middle rows and bottom left: MMD with different values of σ. Bottom right: KS statistic.

According to Figures 3.1 and 3.2, both FGLM and FGLM-MMD are able to achieve a trade-off between predictive accuracy and fairness by alternating the weighting parameter λ in the regression task with lsac dataset. FGLM and FGLM-MMD use $\mathscr{D}_{LC}$ and MMD as fairness penalty, respectively. While FGLM and FGLM-MMD (with σ values equal to or larger than 1) do not show a significant difference in the RMSE-$\mathscr{D}_{LC}$ curves, FGLM-MMD shows slight superiority in the RMSE-MMD curves when σ is small. For example, when $\sigma = 0.01$, there is an area in the plot where the RMSE-MMD curve of FGLM-MMD is under that of FGLM, where that FGLM-MMD has a lower MMD as well as a lower RMSE. KS statistic is a non-parametric test and is useful to measure the difference between the empirical cumulative distribution functions of two continuous one-dimensional distributions. When fairness is measured by KS statistic, FGLM-MMD can achieve much better fairness than FGLM given a similar level of predictive accuracy. In general, FGLM-MMD performs better in terms of *mmd* and KS when σ is small, while it behaves closer to FGLM when σ becomes large. Considering the overall performances of FGLM and FGLM-MMD with different measures of fairness, we conclude that FGLM-MMD provides a flexible trade-off between accuracy and fairness in general.

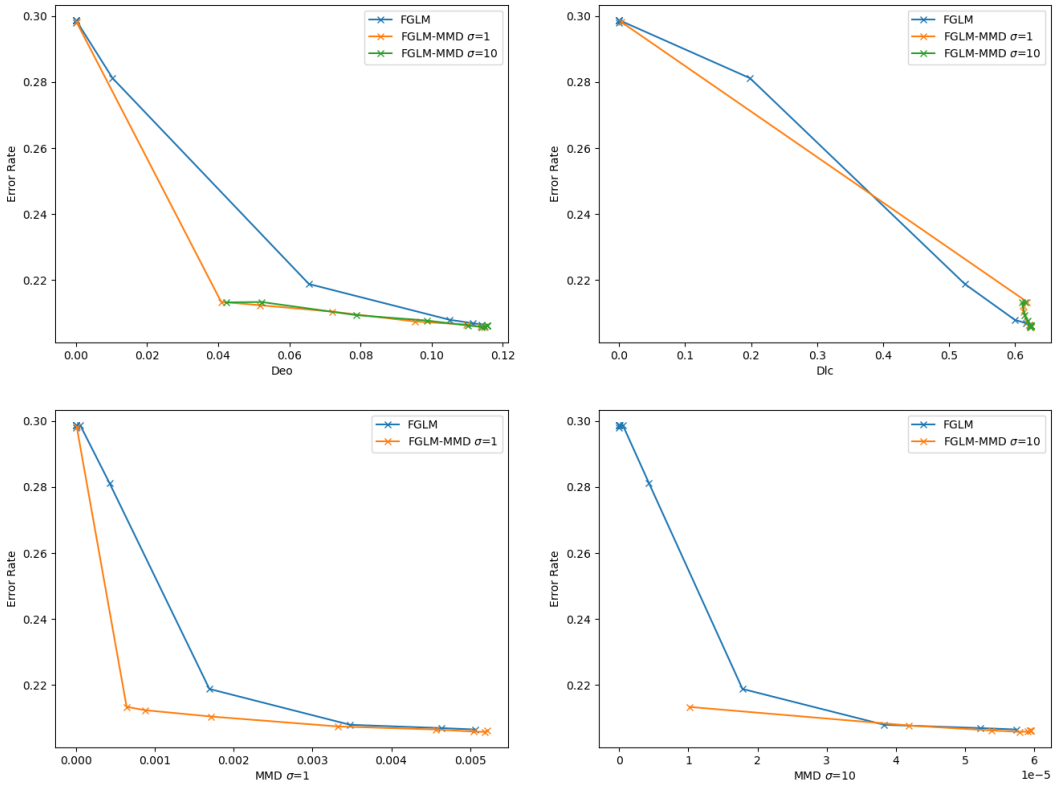

Figure 3.3: Average experimental results for trade-offs of prediction accuracy and fairness on training sets of the German credit dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by predictive error rate, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Bottom left: MMD with $\sigma = 1$. Bottom right: MMD with $\sigma = 10$.

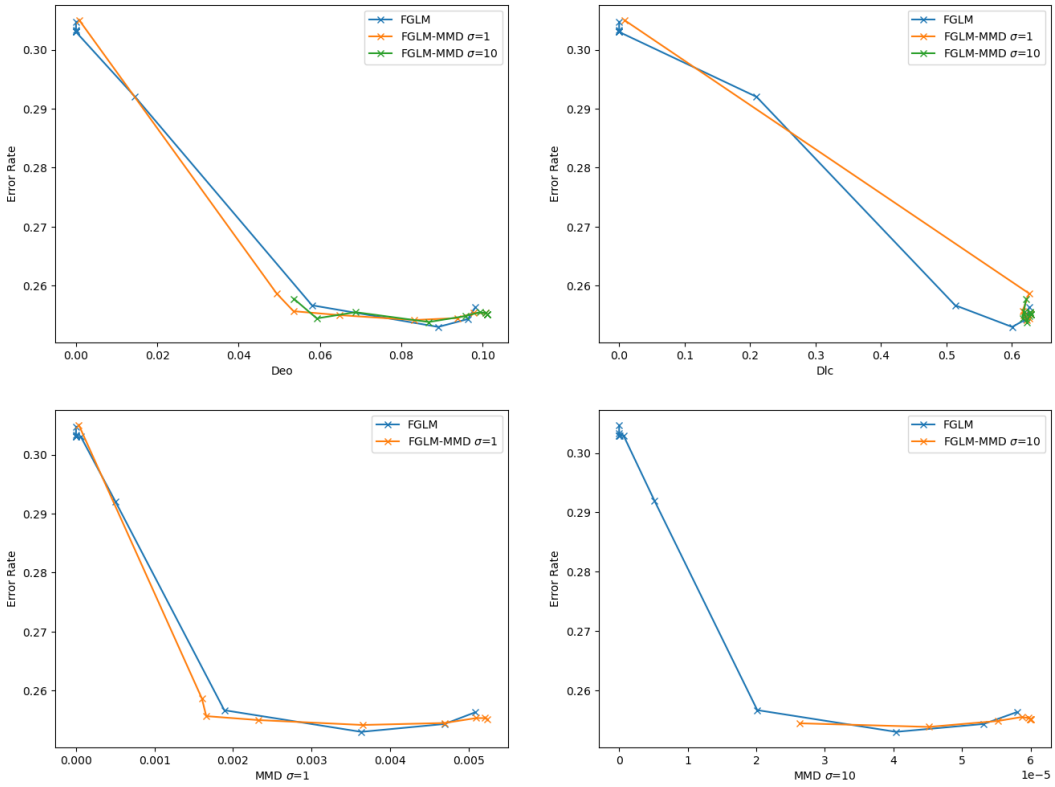

Figure 3.4: Average experimental results for trade-offs of prediction accuracy and fairness on testing sets of the German credit dataset of FGLM and FGLM-MMD with different σ values. Accuracy is measured by predictive error rate, and fairness is measured by different metrics. Top left: $\mathscr{D}_{EO}$. Top right: $\mathscr{D}_{LC}$. Bottom left: MMD with $\sigma = 1$. Bottom right: MMD with $\sigma = 10$.

Figures 3.3 and 3.4 show training and testing results for the binary classification task with German credit dataset. For the training data, we conclude that FGLM performs better in the trade-off between error rate and $\mathscr{D}_{LC}$, while FGLM-MMD gives fairer solutions in the accuracy-MMD trade-offs. When fairness is measured by $\mathscr{D}_{EO}$, FGLM-MMD also provides overall better results. For the testing data, the curves show similar trends as in the training data. However, the graphs suggest that FGLM-MMD may suffer from overfitting in this example.

3.5 Conclusions

In this chapter, we presented a new framework of fairness-aware generalised linear model using maximum mean discrepancy as the fairness penalty. It enjoys the flexibility of GLMs to be able to handle prediction tasks with different outcome types, while maintaining the convexity of the loss function so that the optimisation can be solved efficiently. We believe that using maximum mean discrepancy can help capture the differences between the entire distributions in more details instead of using only the first few moments. The kernel based method has the flexibility to detect more complex and non-linear relationships. Experimental results from both regression and classification examples show that our algorithm performs well based on different metrics for fairness.

Currently our tests are limited to a few regression and binary classification tasks. It will be interesting to test with more datasets, including those with multi-class or count outcomes. It is also not clear how the choice of kernel functions and parameters will have an impact on model performance.

Appendix A

Additional Experimental Results for Fair Liu Model and Fair Liu Model with Sensitive Features

We present additional experimental results for our fair regression models based on Liu-type estimator on the DAG, the lsac, and three additional datasets.

Figure A.1 shows the results on the standardised DAG simulated dataset simulated with $h = 0.2$, including different choices of k values for fLiu and fLiuS models.

Figure A.2 shows the results on the lsac dataset where race is chosen as the sensitive feature, consisting of five classes.

Three additional real datasets are used for experiments. The first additional dataset we use is the student performance data. It records the final-year grades of 649 students of two Portuguese secondary schools. There are 37 predictors including age, weekly study time, guardians' education level, etc. Sex is considered to be the sensitive feature. The second additional dataset we use is the Communities and Crime (crime) dataset. The outcomes are violent crimes per population among 1993 communities in the US from a survey in 1990. We include 95 predictors such as age and income, while others such as community names are excluded. Race is set to be the sensitive feature and has four classes. The third additional dataset we use is the Parkinson's telemonitoring (Parkinson's UPDRS) dataset. We include 16 predictors consisting of measurements form 5875 biomedical voice recordings across 42

people with early-stage Parkinson's disease. Sex is set to be the sensitive attribute where 67% of the instances are male and 33% are female. The target variable to predict is the UPDRS score, a continuous score that evaluates various aspects of Parkinson's disease.

The data split is repeated 20 times and the average accuracy-fairness trade-offs of FGLM, FRRM and fLiu/fLiuS with different k values for the training and testing sets of all three data are shown in Figures A.3-A.5, respectively. In each figure, accuracy is measured by RMSE and is plotted on the vertical axis and fairness measured by different metrics is shown on the horizontal, both of which are the lower the better.

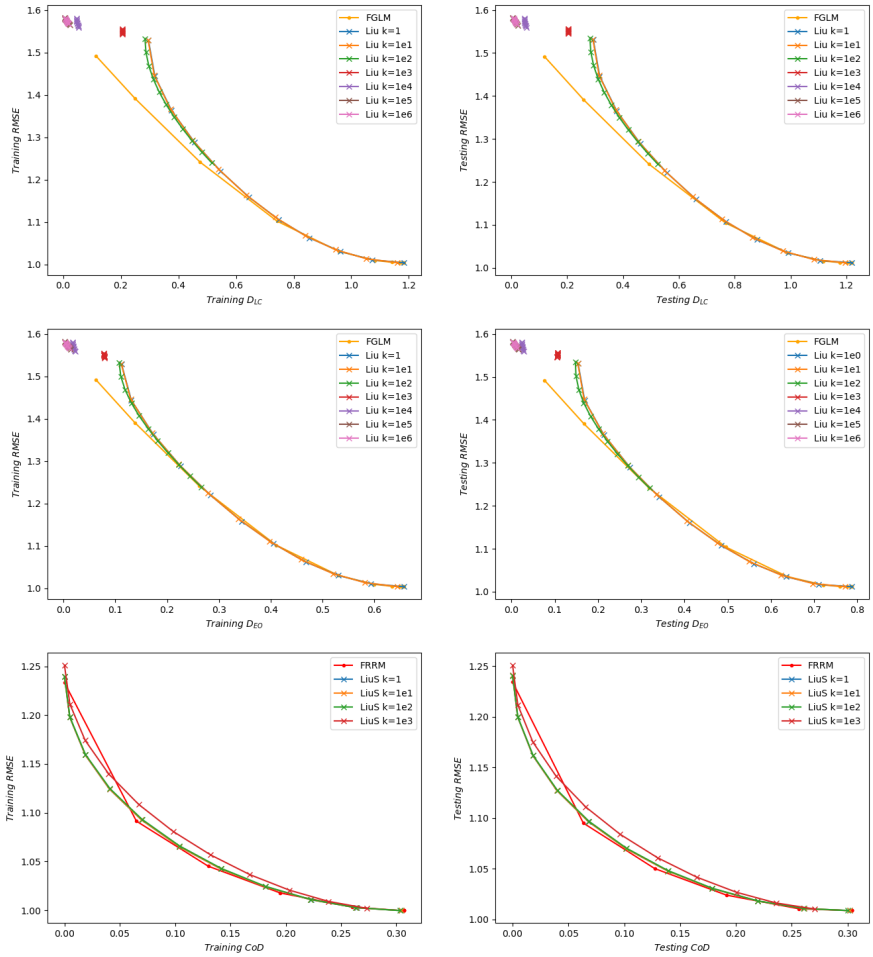

Figure A.1: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the standardised DAG simulated dataset of FRRM, fLiu, FGLM and fLiuS.

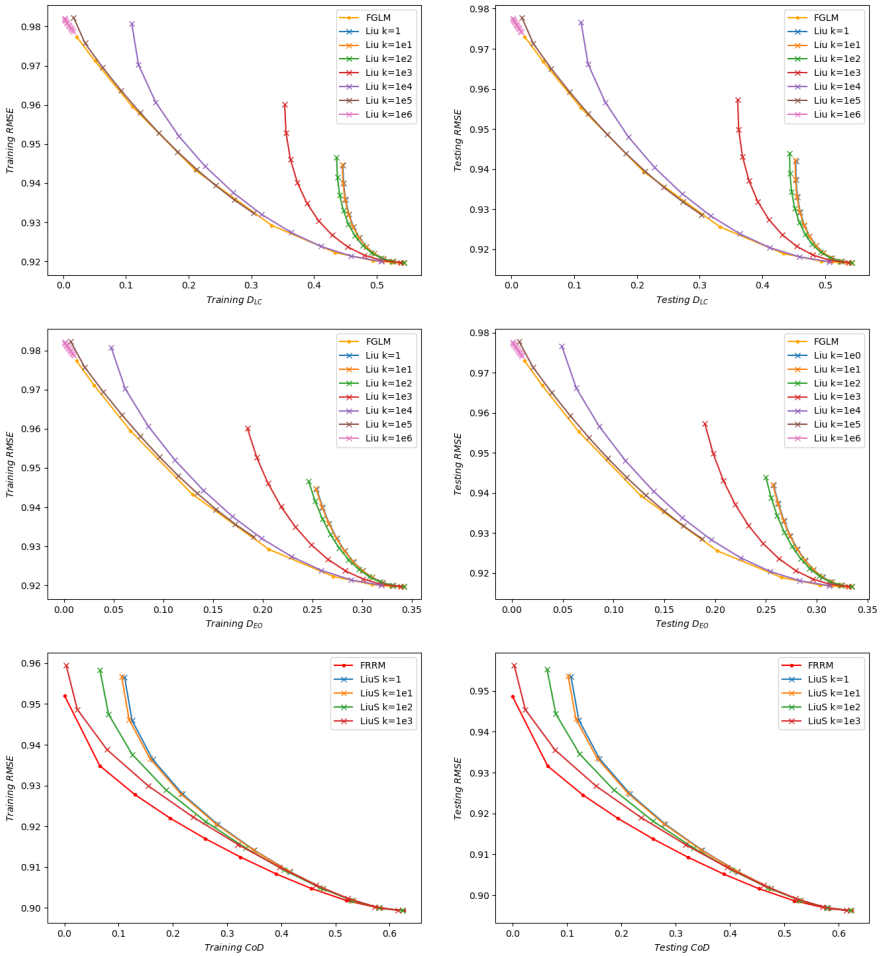

Figure A.2: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the lsac dataset (standardised) with races (Asian, black, Hispanic, white and other) as sensitive feature of FRRM, fLiu, FGLM and fLiuS.

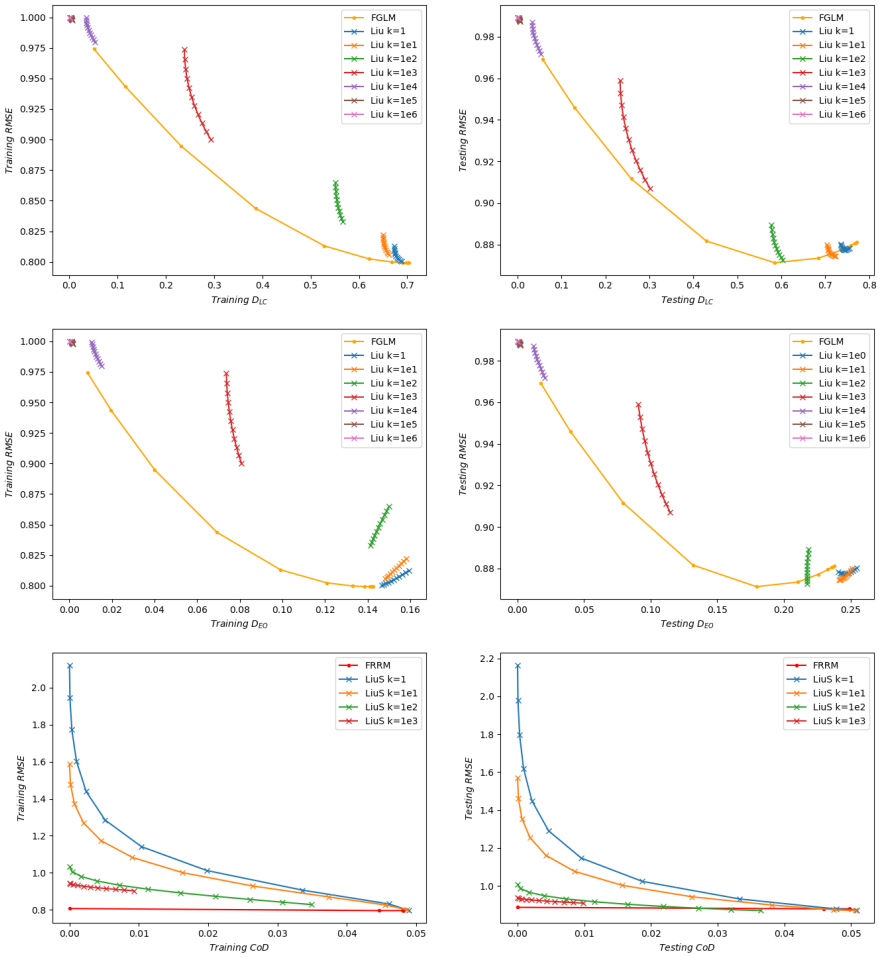

Figure A.3: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the student_performance dataset (standardised) with black versus non-black race as sensitive feature of FRRM, fLiu, FGLM and fLiuS.

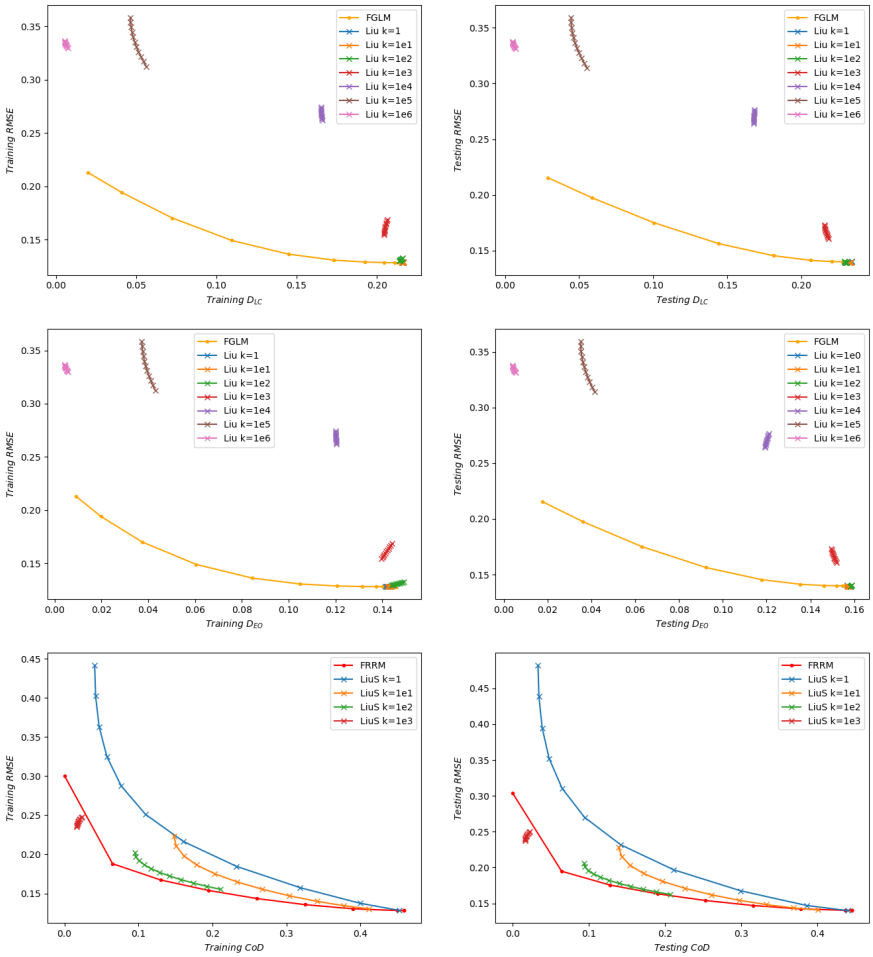

Figure A.4: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the crime dataset (standardised) with black versus non-black race as sensitive feature of FRRM, fLiu, FGLM and fLiuS.

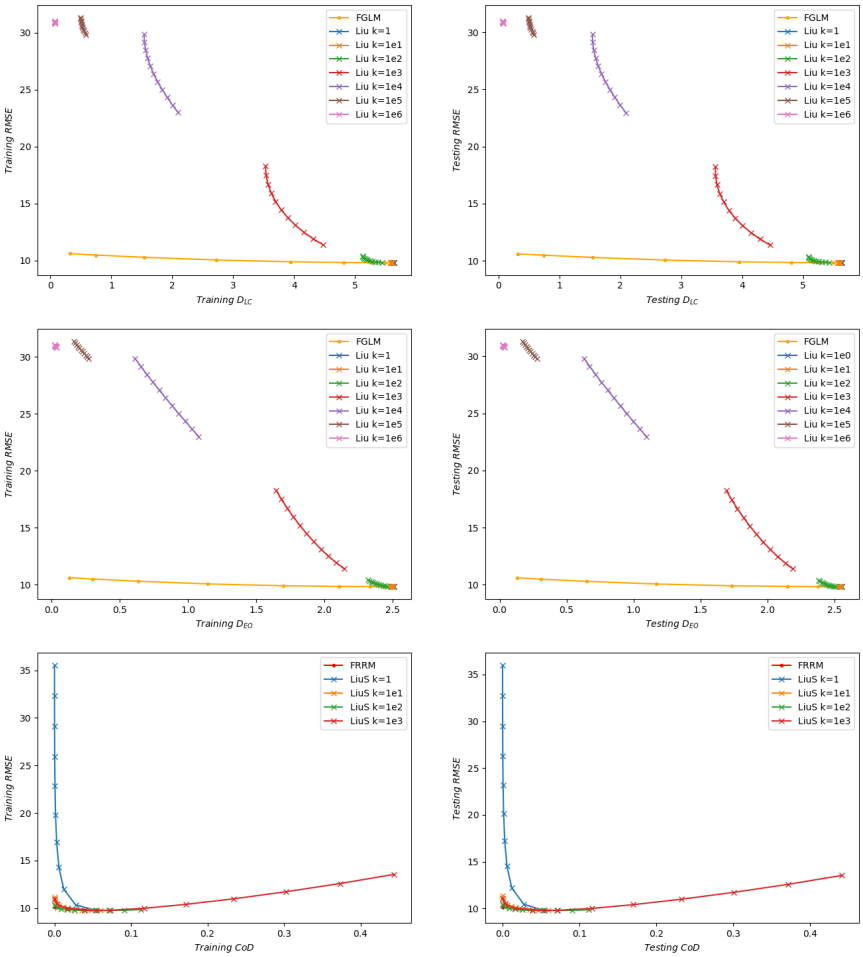

Figure A.5: Experimental results for trade-offs of prediction accuracy (measured by RMSE) and fairness (measured by $\mathscr{D}_{LC}$, $\mathscr{D}_{EO}$ and CoD, respectively) on training and testing sets of the parkinsons_updrs dataset (standardised) with black versus non-black race as sensitive feature of FRRM, fLiu, FGLM and fLiuS.

Bibliography

[1] Xavier Milhaud, Victorien Poncelet, and Clement Saillard. Operational choices for risk aggregation in insurance: Psdization and scr sensitivity. *Risks*, 6(2):36, 2018.

[2] Rainer Opgen-Rhein and Korbinian Strimmer. Accurate ranking of differentially expressed genes by a distribution-free shrinkage approach. *Statistical applications in genetics and molecular biology*, 6(1), 2007.

[3] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[4] Ralph E Steuer, Yue Qi, and Maximilian Wimmer. Computing cardinality constrained portfolio selection efficient frontiers via closest correlation matrices. *European Journal of Operational Research*, 313(2):628–636, 2024.

[5] Stefan Cutajar, Helena Smigoc, and Adrian O'Hagan. Actuarial risk matrices: the nearest positive semidefinite matrix problem. *North American Actuarial Journal*, 21(4):552–564, 2017.

[6] Nicholas J Higham. Computing the nearest correlation matrix—a problem from finance. *IMA Journal of Numerical Analysis*, 22(3):329–343, 2002.

[7] Riccardo Rebonato and Peter Jäckel. The most general methodology to create a valid correlation matrix for risk management and option pricing purposes. *Available at SSRN 1969689*, 2011.

[8] Houduo Qi and Defeng Sun. A quadratically convergent newton method for computing the nearest correlation matrix. *SIAM Journal on Matrix Analysis and Applications*, 28(2):360–385, 2006.

[9] Houduo Qi and Defeng Sun. An augmented lagrangian dual approach for the h-weighted nearest correlation matrix problem. *IMA Journal of Numerical Analysis*, 31(2):491–511, 2011.

[10] Dana Pessach and Erez Shmueli. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, 55(3):1–44, 2022.

[11] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International conference on machine learning*, pages 60–69. PMLR, 2018.

[12] Yahav Bechavod and Katrina Ligett. Penalizing unfairness in binary classification. *arXiv preprint arXiv:1707.00044*, 2017.

[13] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. *Advances in neural information processing systems*, 31, 2018.

[14] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th international conference on world wide web*, pages 1171–1180, 2017.

[15] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

[16] Alekh Agarwal, Miroslav Dudík, and Zhiwei Steven Wu. Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning*, pages 120–129. PMLR, 2019.

[17] Richard Berk, Hoda Heidari, Shahin Jabbari, Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. A convex framework for fair regression. *arXiv preprint arXiv:1706.02409*, 2017.

[18] Luca Oneto, Michele Donini, and Massimiliano Pontil. General fair empirical risk minimization. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[19] Adrián Pérez-Suay, Valero Laparra, Gonzalo Mateo-García, Jordi Muñoz-Marí, Luis Gómez-Chova, and Gustau Camps-Valls. Fair kernel learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 339–355. Springer, 2017.

[20] Hyungrok Do, Preston Putzel, Axel S Martin, Padhraic Smyth, and Judy Zhong. Fair generalized linear models with a convex penalty. In *International Conference on Machine Learning*, pages 5286–5308. PMLR, 2022.

[21] Vali Asimit, Runshi Wang, Feng Zhou, and Rui Zhu. Efficient positive semidefinite matrix approximation by iterative optimisations and gradient descent method. *Risks*, 13(2):28, 2025.

[22] K Pearson. Notes on regression and inheritance in the case of two parents. In *Proceedings of the Royal Society of London*, volume 58, pages 240–242, 1895.

[23] Nicholas J Higham, Natasa Strabic, and Vedran Sego. Restoring definiteness via shrinking, with an application to correlation matrices with a fixed block. *SIAM Review*, 58(2):245–263, 2016.

[24] Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, 88(2):365–411, 2004.

[25] Charles F Van Loan and Gene H Golub. *Matrix computations*. Johns Hopkins University Press Baltimore, 1983.

[26] James P Boyle and Richard L Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. In *Advances in order restricted statistical inference*, pages 28–47. Springer, 1986.

[27] Richard L Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

[28] Frank Deutsch and Hein Hundal. The rate of convergence for the method of alternating projections, ii. *Journal of Mathematical Analysis and Applications*, 205(2):381–405, 1997.

[29] Rüdiger Borsdorf and Nicholas J Higham. A preconditioned newton algorithm for the nearest correlation matrix. *IMA Journal of Numerical Analysis*, 30(1):94–107, 2010.

[30] Amir Ali Ahmadi and Georgina Hall. Sum of squares basis pursuit with linear and second order cone programming. *Algebraic and Geometric Methods in Discrete Mathematics*, 685:27–53, 2017.

[31] Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

[32] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[33] Marco Scutari, Francesca Panero, and Manuel Proissl. Achieving fairness with a simple ridge penalty. *Statistics and Computing*, 32(5):77, 2022.

[34] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

[35] Jiahao Chen, Nathan Kallus, Xiaojie Mao, Geoffry Svacha, and Madeleine Udell. Fairness under unawareness: Assessing disparity when protected class is unobserved. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 339–348, 2019.

[36] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[37] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in neural information processing systems*, 30, 2017.

[38] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2017, 2017.

[39] Junpei Komiyama, Akiko Takeda, Junya Honda, and Hajime Shimao. Nonconvex optimization for regression with fairness constraints. In *International conference on machine learning*, pages 2737–2746. PMLR, 2018.

[40] Kejian Liu. Using liu-type estimator to combat collinearity. *Communications in Statistics-Theory and Methods*, 32(5):1009–1020, 2003.

[41] Liu Kejian. A new class of blased estimate in linear regression. *Communications in Statistics-Theory and Methods*, 22(2):393–402, 1993.

[42] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 560–568, 2008.

[43] Simon Caton and Christian Haas. Fairness in machine learning: A survey. *ACM Computing Surveys*, 56(7):1–38, 2024.

[44] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P Gummadi. Fairness constraints: A flexible approach for fair classification. *Journal of Machine Learning Research*, 20(75):1–42, 2019.

[45] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[46] Marvin Gruber. *Improving efficiency by shrinkage: The James–Stein and Ridge regression estimators*. Routledge, 2017.

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

[48] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24-28, 2012. Proceedings, Part II 23*, pages 35–50. Springer, 2012.

[49] Luca Oneto, Michele Donini, Giulia Luise, Carlo Ciliberto, Andreas Maurer, and Massimiliano Pontil. Exploiting mmd and sinkhorn divergences for fair and transferable representation learning. *Advances in Neural Information Processing Systems*, 33:15360–15370, 2020.