Kernel Methods: Generalisations, Scalability and Towards the Future of Machine Learning



Jack Fitzsimons

The Queen's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2019

Acknowledgements

I would first like to thank my supervisors Prof Michael Osborne and Prof Stephen Roberts for their belief in me and continued support. They have not only shown a great deal of patience but have also provided an environment in which I have been able to develop as an independent researcher with the necessary support along the way. I am confident that their roles in my graduate education will stand to me for many years to come.

This work would also not have been possible without the strong collaborators I've had over the past few years, such as: Prof Maurizio Filippone, Prof Joseph Fitzsimons, Dr Zhikuan Zhao, Kurt Cutajar and Diego Granziol. Their insightful conversations and enthusiasm led to each of the publications at the core of this thesis.

Of course, not all conversations and inspired moments translate into published work. I am deeply indebted to all the members of the Machine Learning Research Group at the University of Oxford for the numerous brainstorms at whiteboards, thought-provoking discussions at the Crown Victoria and social outings. A special thank you goes to Dr Steve Reece, Jonathan Downing, Logan Graham, Dr Justin Bewsher, Dr Ibrahim Almosallam, Dr Tom Nickson, Dr Elmarie van Heerden, Rory Beard, Dr Tom Gunter, Dr Chris Llyod, Dr Tom Rainforth, Gabriele Abbati, Dr Glen Calopy, Dr Ali Syed Rizvi and Dr Favour Nyikosa.

Finally and most importantly, I am forever thankful to my family, especially to my parents. They have not only supported me through my graduate education but have shown my siblings and I the importance of education and hard work since a young age. They have acted as great role models to us all.

Viva Examiners

I would like to formally thank my viva examiners,

Prof Carl Rasmussen, Machine Learning Group, Department of Engineering, University of Cambridge.

Dr Jan-Peter Calleiss, Machine Learning Research Group, Oxford-Man Institute for Quantitative Finance, University of Oxford.

for their interesting discussion, insightful comments and advice for future research. I hope to carry their advice and comments with me in my future research endeavours.

Abstract

Kernel methods are a broad class of machine learning algorithms made popular by Gaussian processes and support vector machines. Other popular methods, less commonly referred to as kernel methods, are decision trees, neural networks, determinantal point processes and Gauss Markov random fields.

There are three core areas of contribution in this thesis, namely the generalization, scalability and future of kernel methods.

The work begins by introducing kernel methods from the viewpoint of regression and classification, identifying the links between a myriad of machine learning algorithms. This general perspective of kernel methods is leveraged to address an important question faced by our field; how do we develop machine learning techniques which constrains inference as to maintain equality in expectation of the predictor. This has application to an open problem in algorithmic fairness referred to as Group Fairness, that is to do not suffer from group level inequalities. To answer this, a novel definition for statistical parity (group fairness in expectation) is introduced and is shown to have a natural form as constrained kernel regression. A key feature of this fairness constraint is that it can easily be incorporated into many widespread models in production without the requirement of retraining. It also deals with issues such as intersectionality and is applied to synthetic data, salary predictions of civil servants in the state of Illinois and the ProPublica dataset.

Scalability is the second core focus of the work. Kernel methods are often un-

able to be solved for in closed form when dealing with large datasets. This is due to the algorithmic bottleneck of the necessary linear algebraic operations. While there has been ample work on approximating kernel matrices via low-rank approximations, Bayesian committee machines and hierarchical matrices to name but a few approaches, little work has been investigated in the area of stochastic trace estimation. This work thoroughly explores this avenue. It starts with developing a novel technique for sampling the probe vectors required for stochastic trace estimation which provably improves on the previous state-of-the-art approach.

Following this, both Bayesian and entropic approaches to infer important linear algebraic operations such as log-determinants are considered. These offer a novel perspective of the linear algebraic operations as the inference of an unknown distribution of eigenvalues. As such, stochastic trace estimation provides noisy observations of the moments of the distribution of eigenvalues. These two novel methods are shown to improve over the previously popular Taylor and Chebyshev approximations and the stochastic Lanczos method as applied to the UFL sparse dataset, with application to Gauss Markov random fields, determinantal point processes and Gaussian processes.

Finally, the future of kernel methods is considered. Over recent years, an extraordinary push to develop quantum computers has been made in both academic and commercial environments. It appears undoubtable that quantum computers will play a significant role in the future of linear algebraic heavy algorithms. This is explored in the context of Gaussian processes, with a novel set of quantum algorithms which run in polylogarithmic time. While the results are mainly theoretical for now, the potential to study broader machine learning algorithms is discussed.

Contents

I Machine Learning: The State of Affairs **1**

1 Introduction **2**

II Kernel Methods as a Way of Life **10**

2 Fitting Lines **11**

2.1 Linear Regression and Logistic Classification 12

2.1.1 Ordinary Least Squares 12

2.1.2 Under- and Over-Complete Regression Problems 13

2.1.3 Bayesian Linear Regression 14

2.1.4 Frequentist Linear Regression 16

2.1.5 Classification 18

2.2 Conclusion 20

3 The Kernel Trick **21**

3.1 Design Matrices 21

3.2 The Kernel Trick 22

3.2.1 Common Kernels 23

3.2.2 Kernel Combination 25

3.3 Kernel Methods in Machine Learning 27

3.3.1 Gaussian Processes . . . 27
3.3.2 Support Vector Machines . . . 28
3.3.3 Gauss Markov Random Fields . . . 29
3.3.4 Determinental Point Processes . . . 29
3.4 Links to Other Machine Learning Models . . . 30
3.4.1 Decision Trees and Random Forests . . . 30
3.4.2 Neural Networks . . . 31
3.5 Conclusion . . . 32

4 Constrained Kernel Regression with Applications to Group Fairness 33
4.1 Related Work . . . 36
4.2 Constrained Kernel Regression . . . 39
4.3 Trees As Kernel Regression . . . 41
4.4 Fairness Constrained Decision Trees . . . 43
4.5 Efficient Algorithm For Equality Constrained Decision Trees . . . 45
4.5.1 Compressed Kernel Representation . . . 45
4.5.2 Explicit Kernel Representation . . . 47
4.6 Expected Perturbation Bounds . . . 48
4.7 Combinations Of Fair Trees . . . 51
4.8 Experiments . . . 51
4.8.1 Synthetic Demonstration . . . 51
4.8.2 ProPublica Dataset - Racial Biases . . . 52
4.9 Intersectionality . . . 55
4.10 Experiments: Intersectionality . . . 56
4.10.1 ProPublica & the COMPAS System . . . 56
4.10.2 Illinois State Employee Salaries . . . 57

4.11 Conclusion 58

III Overcoming Scalability Issues 60

5 Scaling Gaussian Processes 61

5.1 Parametric Sparse Gaussian Processes 61

5.1.1 Subset of the Data Points (SoD) 64

5.1.2 Nyström's Method 64

5.1.3 Subset of Regressors (SoR) 66

5.1.4 Deterministic Training Conditional (DTC) 66

5.1.5 Fully/Partially Independent Training Conditional (F/PITC) . 68

5.1.6 Random Fourier Features 69

5.2 Local Expert Models 69

5.2.1 Multiplicative methods 70

5.2.2 Treed Gaussian Processes 73

5.3 Other Methods 74

5.3.1 Toeplitz Covariance Matrices 74

5.3.2 Hierarchical Matrices and Fast Multipole Methods 75

5.4 Conclusion 76

6 Implicit Matrix Approximation 77

6.1 Stochastic trace estimation 79

6.2 Mutually unbiased bases 80

6.3 A novel approach to trace estimation 83

6.3.1 Analysis of Fixed Basis Estimator 85

6.3.2 Analysis of MUBS Estimator 86

6.4 Experimentation . . . 89
6.4.1 Theoretical Resuls . . . 89
6.4.2 Numerical Result . . . 89
6.4.2.1 Example Matrix . . . 89
6.4.2.2 Counting Triangles in Graphs . . . 90
6.4.2.3 Log Determinant of Covariance Matrix . . . 92
6.5 Conclusion . . . 94

7 Probabilistic Numeric Eigen-Spectrum Inference 96
7.1 Introduction . . . 96
7.2 Background . . . 99
7.2.1 Taylor Approximation . . . 100
7.3 A Probabilistic Numerics Approach . . . 101
7.3.1 Raw Moment Observations . . . 102
7.3.1.1 Bayesian Quadrature . . . 103
7.3.2 Inference on the Log-Determinant . . . 104
7.3.2.1 Histogram Kernel . . . 104
7.3.2.2 Polynomial Kernel . . . 106
7.3.2.3 Prior Mean Function . . . 107
7.3.2.4 Bai & Golub Log-Determinant Bounds . . . 108
7.3.2.5 Using Bounds on the Log-Determinant . . . 108
7.3.2.6 Algorithm Complexity and Recap . . . 109
7.4 Experiments . . . 110
7.4.1 Synthetically Constructed Matrices . . . 113
7.4.2 UFL Sparse Datasets . . . 114
7.4.3 Uncertainty Quantification . . . 115

7.4.4 Motivating Example . . . 116
7.5 Conclusion . . . 117

8 Entropic Eigen-Spectrum Inference **119**
8.1 Raw Moments of the Eigenspectrum . . . 120
8.2 Approximating the Log Determinant . . . 121
8.3 Estimating the Log Determinant using Maximum Entropy . . . 122
8.3.1 Implementation . . . 123
8.4 Experiments . . . 125
8.4.1 UFL Datasets . . . 125
8.4.2 Computation of GMRF Likelihoods . . . 127
8.5 Conclusions . . . 129
8.6 Sparse Linear Algebra for Gaussian Processes . . . 131

IV Towards the Future of Machine Learning **134**

9 Quantum Computing for Machine Learning **135**
9.1 The Qubit . . . 136
9.2 Quantum Random Access Memory (QRAM) . . . 137
9.3 Phase Estimation . . . 138
9.4 Quantum Gaussian Process Training and Inference . . . 139
9.5 Application to Other Machine Learning Models . . . 141
9.6 Conclusion . . . 141

10 Conclusion **143**

Bibliography **147**

List of Figures

1.1 Public popularity of machine learning algorithms. 4
1.2 Changes in popularity of machine learning techniques. 5

3.1 Kernels for Gaussian Processes. 26

4.1 Visualization of constrained kernel matrices. 44
4.2 Fair regression applied to diverse models. 53
4.3 Fair regression applied to the ProPublica dataset. 54
4.4 Intersectionality constraints on ProPublica. 57
4.5 Illinois salary prediction with fairness constraints. 58

5.1 Visual comparison of Bayesian Committee Machine Variants. 73
5.2 Visualisations of hierarchical matrices. 75

6.1 Convergence of Stochastic Trace Estimation on spiked data. 91
6.2 Convergence of Stochastic Trace Estimation on low rank data. 91
6.3 Benchmark of Stochastic Trace Estimation techniques. 93
6.4 Comparison of STE methods in estimating log determinants. 95

7.1 Theoretical error of Taylor series approximations of kernel matrices. . 99
7.2 Comparison of log determinant estimates via different STE techniques.111
7.3 Comparison of trace estimation approaches on the UFL dataset. . . . 114
7.4 Quality of uncertainty estimates for Bayesian trace estimation. 115

7.5 BILD estimates to training DPPs. 117

8.1 UFL Dataset with Entropic trace estimates. 125

8.2 Speed of training GMRFs using ELD estimates. 128

8.3 Accuracy of ELD estimates in log likelihoods. 129

8.4 The image on the left is a visualisation of the dot product kernel when all hyper param

9.1 Above is a visual diagram of a single qubit taken from Wikipedia. . 137

Part I

Machine Learning: The State of Affairs

Chapter 1

Introduction

This thesis, as the title may have suggested, is focussed on the field of machine learning. Beyond the misleading news headlines, the term *machine learning* is generally attributed to Arthur Samuel in 1959 [140]. Arthur described machine learning as the field of artificial intelligence which used statistical techniques to allow computer programs to progressively improve at a task with experience (data). Tom Mitchell[1] provided a more formal definition of machine learning as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

Today, the field of machine learning has greatly matured from its early days, becoming pervasive throughout society. From the choice of advertisement on a website through to weather forecasting, product placements and recommendations, financial modeling, call center optimization, cyber security, gaming and an ever-growing list of industries and products, machine learning has been pushing the state of the art. As such, research both theoretical and applied has snowballed with an ever increasing number of submissions to NIPS, ICML, UAI, AISTATS,

[1] Tom's book "Machine Learning" was the first machine learning book I read and got me interested in the field originally.

AAAI and so on.

Research is, of course, contextual, especially in emerging fields such as machine learning. Before diving into the body of the of the thesis it is important to see contextually where this work fits in the broader domain of machine learning. The work presented here focusses on kernel methods, as introduced in Chapter 3. Kernel methods are only one piece of the broader machine learning landscape and it would be ignorant to not acknowledge some of the others.

As Prof Neil Lawrence described at the Machine Learning Summer School in 2015, machine learning has been a lot like a self-eating snake whereby the topics of research in vogue have iterated from neural networks to restricted Boltzmann machine, to tree methods, to kernels and now back to neural networks. While the topics under most consideration cycle, I believe it is far more important to focus on the similarity between techniques rather than their differences. Not confining one's research to a single domain but rather to embrace the benefits and differences of each. As such, while the focus of the thesis is on kernel methods there has been a conscious effort to bridge the gap between, for example, Gaussian processes and decision trees in this thesis. In doing so, interesting practical novel approaches have been created.

Ultimately as a researcher I've had to ask myself many times *why* I am doing this work. In my transfer of status, Prof Frank Wood gave me the powerful advice "do something worth doing." While this may sound obvious I do believe it is good advice. So what is it that people care about and what seems like a reasonable direction for research? If we can answer this half of the work is already done.

Let's start by looking at what people care about. Figures 1.1 and 1.2 display the

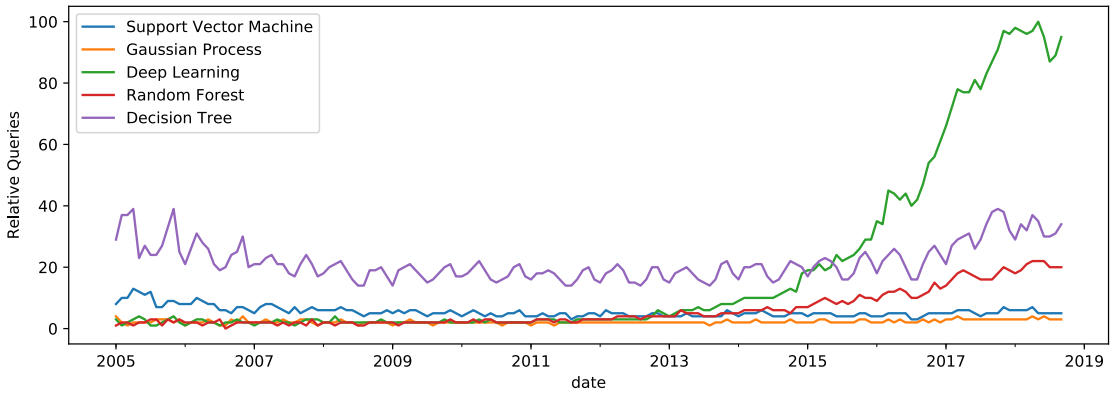


Figure 1.1: The figure shows the relative number of Google search queries for some popular machine learning approaches.

relative number of Google queries[2] for a number of machine learning approached and the relative change of Google queries for a range of terms related to broader artificial intelligence and data analytics. Starting with Figure 1.1, it can be clearly seen the huge growth in popularity around deep learning models. This is no surprise as it can both be seen in the number of paper submissions to top conferences and also the amount of media attention that deep learning has been receiving over the past number of years. In fairness to the deep learning community, they are deserving of the recognition as they have dramatically changed our ability to do image recognition, machine translation and speech recognition. Figure 1.1 tells us a more interesting story than this. If one looks closely one can see a dip in the use of support vector machines (SVMs). Once used as the classification technique of choice by the computer vision community, who generally used SIFT features and other handcrafted features, they have now drastically moved over to the deep learning style of inference. Further, we can see Random Forests growing rapidly

[2]Found using the Google trends API

in popularity. Practitioners competing in machine learning competitions, such as those on Kaggle, have showcased the effectiveness of ensemble methods. Today many *data science* courses are seen to not be complete without covering decision trees and ensemble approaches due to their simple theory and empirical effectiveness. Interesting to note is the oscillation in popularity for decision trees. It would appear that their popularity may be down to how often they are taught to students as term times oscillate in a similar pattern.

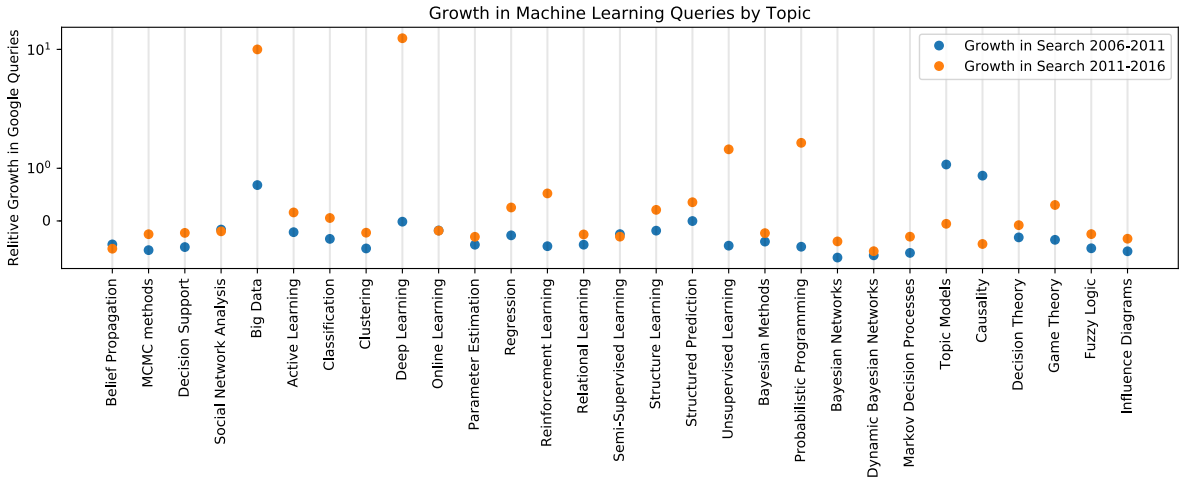


Figure 1.2: The figure shows the relative change of Google search queries for some terms related to machine learning and data analytics.

Figure 1.2 highlights some of the key topics that stand out as growing in popularity over the last decade. Big data and deep learning, beyond any debate, have surged in interest. Unsupervised learning and probabilistic programming have begun to grow, followed by reinforcement learning regression and game theory. As explained in the following chapters, regression can be seen as the backbone of classification and thus has always appealed to my interests more strongly.

Big data, a term often used by practitioners when the scale of data they operate is orders of magnitude large than traditional application, and deep learning have

a lot in common. For a task to run on big data, generally speaking, it would have to have close to $\mathcal{O}(n)$ computational and memory complexity, where n is the size of the dataset. Deep learning, generally speaking, requires large datasets to learn useful feature spaces. Gaussian processes, for example, do not scale naively to large datasets. Thinking of how to scale Gaussian processes primarily for regression to large datasets has become a predominant theme in this thesis. There is also a very strong link between Gaussian processes, decision trees and neural networks as they are all essentially forms of kernel method. This thesis premises the Gaussian processes may be every bit as powerful as, for example, neural networks if we can alleviate the computational bottle necks. This thesis thus focusses on the usage of Gaussian processes, the scalability of Gaussian processes to large datasets and the future of Gaussian processes by taking advantage of quantum computers.

The thesis is laid out as follows;

[Chapter 2] introduces how we think about fitting lines to data from both a Bayesian and frequentist viewpoint.

[Chapter 3] introduces design matrices and the kernel trick, linking kernels more broadly to a range of machine learning methods.

[Chapter 4] leverages similarities between kernel methods and decision trees to tackle an important issue facing the community: fairness in machine learning.

[Chapter 5] surveys the literature regarding scaling kernel methods.

[Chapter 6] examines the linear algebraic bottleneck for solving log determinants and offers an improved stochastic trace estimation approach.

[Chapter 7] leverages the approach outlined in Chapter 6 and offers a Bayesian

approach to log determinant calculations.

[Chapter 8] improves on Chapter 7 by refining the Bayesian approach with an entropy framework, offering improved empirical results.

[Chapter 9] looks towards the future of Gaussian processes and kernel methods more generally as they are applied on quantum computers.

Through this work, eight research papers were contributed to. Chapters which follow the work of individual or multiple papers follow the notations in that paper in order to give readers of the papers more context. However, as a consequence, this means some of the notation is inconsistent between chapters. As such each chapter is self-contained and introduces notation as appropriate. The research contributions have been;

[1] Jack Fitzsimons, Abdulrahman Al Ali, Michael Osborne, and Stephen Roberts. Group Fairness Constraints for Decision Trees and Related Methods, Special Issue on Entropy Based Inference and Optimization in Machine Learning, *Entropy*, 2019.

I identified the open problem, developed the theory, wrote and ran the experiments, wrote the initial draft of the paper, gathered feedback and edited the paper.

[2] Jack Fitzsimons, Michael Osborne, and Stephen Roberts. Intersectionality: Multiple Group Fairness in Expectation Constraints. *NeurIPS Workshop on Ethical, Social and Governance Issues in AI 2018*.

I identified the problem, developed the theory, wrote and ran the experiments, wrote the initial draft of the paper, gathered feedback and edited the paper.

[3] Jack Fitzsimons, Kurt Cutajar, Michael Osborne, Stephen Roberts, and Maurizio Filippone. Bayesian inference of log determinants. In *Uncertainty in Artificial*

Intelligence 2017.

I identified the problem, developed the core theory, wrote and ran the experiments and helped to edited the paper.

[4] Jack Fitzsimons, Diego Granziol, Kurt Cutajar, Michael Osborne, Maurizio Filippone, and Stephen Roberts. Entropic trace estimates for log-determinants. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 323 338. Springer, 2017.

I identified the problem, jointly developed the core theory, wrote and ran the experiments and helped to edited the paper.

[5] Jack K. Fitzsimons, Michael A. Osborne, Stephen J. Roberts, and Joseph F. Fitzsimons. Improved Stochastic Trace Estimation using Mutually Unbiased Bases. *Uncertainty in Artificial Intelligence 2018*.

I identified the problem, jointly developed the core theory, wrote and ran the experiments, drafted the paper and helped to edited the paper.

[6] Zhikuan Zhao, **Jack K. Fitzsimons**, and Joseph F. Fitzsimons. Quantum-Assisted Gaussian Process Regression, *Physical Review A*, 2019.

I jointly identified the problem, jointly developed the initial theory and helped to edited the paper.

[7] Zhikuan Zhao, **Jack K. Fitzsimons**, Michael A. Osborne, Stephen J Roberts, and Joseph F. Fitzsimons. Quantum Algorithms for Training Gaussian Processes, *Physical Review A*, 2019.

I jointly identified the problem, jointly developed the initial theory and helped to edited the paper.

[8] Zhikuan Zhao, Vedran Dunjko, **Jack K. Fitzsimons**, Patrick Rebentrost and Joseph F. Fitzsimons. A Note on State Preparation for Quantum Machine Learning. Under Review.

I was part of core idea creation and helped with minor edits.

Part II

Kernel Methods as a Way of Life

Chapter 2

Fitting Lines

While it's not uncommon to associate fitting a line to points with high school maths, I would argue that it is the most fundamentally important task of modern statistics and machine learning.

Sir Francis Galton is generally credited for introducing linear regression during his investigation into genetics and heredity at the end of the 19^{th} century [148][1]. The motivating work, as described in his biography [122], was to understand the relationship between the weight of a sweet pea seed and that of its daughter seeds after pollination. He found that if he plotted the median weight of the daughter seeds with respect to the weight of the mother seed, the data traced out a straight line. Finding such linear models to describe the relationship between variables is still performed to this day in finance, medicine and throughout the sciences.

Pearson, a colleague of Galton, noted that the optimal regression slope and correlation coefficient could be found by the product sum formula under least squares error. This early understanding of linear regression is now referred to as *ordinary least squares* and that, in my opinion, is where the story of machine learning begins.

[1] Unfortunately, many feel that the fame of Galton was overshadowed by that of his cousins, Charles Darwin. One can only imagine having a close relative with greater scientific standing!

2.1 Linear Regression and Logistic Classification

2.1.1 Ordinary Least Squares

Two important aspects arise from fitting lines to points. Statisticians are generally interested in understanding the relationship between the input x and the output y as it may tell us something interesting about the system. The machine learning community is concerned more with using the line we have deduced in order to interpolate and extrapolate away from the set of observations and predict behaviours not yet observed.

Given a set of n input-output pairs $\{\mathbf{x}_i, y_i\}_i$, ordinary least squares (OLS) endeavours to find the $\boldsymbol{\beta}$, the vector of weights, which minimises $\sum_i (y_i - \boldsymbol{\beta}^T \mathbf{x}_i)^2$. Assuming that the arithmetic mean of both input and output is zero, Pearson showed that in the one dimensional setting the optimal β, the scalar weight, may be found by the product sum formula, $\beta_{opt} = \sum_i \frac{x_i y_i}{\sum_i x_i^2}$. This can be easily proven by letting Q be the squared error and taking the derivative with resect to β,

$$Q = \sum_i (y_i - \beta x_i)^2$$

$$\frac{dQ}{d\beta} = 2 \sum_i (y_i - \beta x_i)\, x_i := 0 \quad \text{(single local minimum)}$$

$$\sum_i y_i x_i = \beta \sum_i x_i^2 \quad \square$$

This can be extended to the higher dimensional case by extending the above formula using simple linear algebra, $\boldsymbol{\beta} = (X^T X)^{-1} X \mathbf{y}$, where X denotes the concatenation of $\{\mathbf{x}_i\}_i$ into a matrix.

There are very good reasons why regression didn't stop with OLS, to list but a few:

- If the number of observations is less than the number of parameters then the problem is *under-complete* and there exists an infinite space of linear functions which fit the data equally well.
- The least squares or L_2-loss is not always appropriate for the given task.
- There may be a number of outliers in the data set.
- Linear functions may model the relationship between x and y poorly.

What machine learners and statisticians do have generally in common is that they can both be categorised into two groups, namely frequentist and Bayesian. At a high level, frequentists generally consider observations as statistically independent trials of some infinite sequence of possible results with unknown variables treated as fixed but unknown quantities and not as random variables. The Bayesian, on the other hand, considers probabilities over all quantities with which they are uncertain. Bayes' rule formulates how to update any prior beliefs or probabilities based on evidence which is observed from the data. While these two approaches sound subtle at first, the two communities tend to be highly polarised and discussions around which approach to take become as personal as religious or political views. For the sake of appealing to both audiences during this section, I will discuss both frequentist and Bayesian linear models.

2.1.2 Under- and Over-Complete Regression Problems

In the previous subsection we dealt with the problem of over-complete regression, that is to say we observe more observations of data that we have dimensions of X.

However, this is not always the case, especially as we move to high dimensional settings as seen in the next section. In such a scenario, where the dimensionality of X is greater than the number of observations, we need a way of choosing which of the infinite potential fits are best. Under a Bayesian setting, we place a prior on the parameters of the model and update these based on the the evidence observed from the data. As such a lack of data simply leads to a greater reliance on the prior. As low probability mass of the prior is usually placed on complex functions in some sort of entropy based definition of complexity such as Kolmogorov complexity, the prior acts as a natural regulariser. This is especially seen in Gaussian processes as introduced in the next section.

Under a frequentist setting there is no use of priors. Instead *loss functions* are defined. These loss functions are created in order to penalise models which seem less plausible. From Occam's razor, it is assumed that the simplest model which explains the data is best. As such loss functions are generally created in order to minimize model complexity and maximize fit.

2.1.3 Bayesian Linear Regression

The Bayesian philosophy of machine learning and statistics uses probability distributions to encode the uncertainty we have about the world. Bayes' theorem, named after Thomas Bayes, lays out the formulation for updating our initial (prior) probability based on the evidence we observe about a variable,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where $P(A)$ and $P(B)$ are the probability of observing A and B respectively and $P(A|B)$ and $P(B|A)$ are respectively the conditional probabilities of observing A

or B conditioned on the other. There are two key features worth noting. Firstly, each element of Bayes' theorem is encoded as a probability distribution and hence uncertainty is maintained throughout. Secondly, we must initially denote how uncertain we are about the system we are modelling *a priori*.

Some key Bayesian terms which will be used throughout this thesis are:

Prior The distribution assumed before any observations are seen.

Likelihood The distribution associated with the observation obtained.

Posterior The updated distribution formed by combining the prior and likelihood via Bayes' theorem.

Conditional The probability distribution of a variable when the dependent variable is known.

Marginal The distribution found by averaging over all dependent variables.

MLE The maximum of the likelihoods probability density.

MAP The maximum of the Posterior probability density.

For Bayesian linear regression, we assume that the observations are independent of one another given the model, or in the linear case the unique parameters $\boldsymbol{\beta}$. The prior over $\boldsymbol{\beta}$ is typically taken to be a multivariate normal distribution, with zero mean and unit variance which is a reasonable prior[2] after normalisation of the data,

$$p(\boldsymbol{\beta}) = \mathcal{N}(0, \Sigma_{\boldsymbol{\beta}}).$$

[2] A reasonable prior may sound somewhat ambiguous and generally speaking that is because it is. The aim of the prior is to encode the prior beliefs on the system, however, if these beliefs are wildly incorrect then with limited data the prior may be detrimental to the inference.

The likelihood is the probability that the y values were generated given $\boldsymbol{\beta}$. We assume that there is some independent identically distributed Gaussian noise σ_n^2 on each observation. As the observations are assumed to be independently observed, the likelihood factorises as,

$$p(y|\mathbf{x},\boldsymbol{\beta}) = \prod_i p(y_i|\mathbf{x}_i,\boldsymbol{\beta}) = \mathcal{N}(\mathbf{x}\boldsymbol{\beta},\sigma_n^2\mathcal{I}).$$

Combining the prior and likelihood, we can find the posterior distribution over $\boldsymbol{\beta}$,

$$p(\boldsymbol{\beta}|x,y) = p(y|x,\boldsymbol{\beta})p(\boldsymbol{\beta})(\int p(y|x,\boldsymbol{\beta})d\boldsymbol{\beta})^{-1} = \mathcal{N}(\sigma_n^{-2}A^{-1}\mathbf{x}y, A^{-1}),$$

where $A = \sigma_n^{-2}\mathbf{x}\mathbf{x}^T + \Sigma_{\boldsymbol{\beta}}^{-1}$.

An interesting characteristic of Bayesian linear regression, or Bayesian methods more generally, is that we can draw the parameters $\boldsymbol{\beta}$ before and after our observations have been included into the model, seeing how the distribution of $\boldsymbol{\beta}$ adapts to new points and hence the induced functions.

While it is certainly valid to consider non-Gaussian priors on the parameters $\boldsymbol{\beta}$ it is certainly not trivial to find distributions which have such elegant analytic forms, allowing for a tractable posterior to be derived. These are often referred to as conjugate priors. This makes closed form Bayesian methods less flexible than their frequentist counterparts which are not necessarily restricted to the same probabilistic update, unless of course approximate inference is performed.

2.1.4 Frequentist Linear Regression

In the frequentist framework, it does not make sense to talk about probabilities over the unknown variables $\boldsymbol{\beta}$. Instead, $\boldsymbol{\beta}$ is estimated by more crude interpretation

of Occam's razor, that is to use the simplest hypothesis which explains the data. Of course, the concept of *simple* is ambiguous but many interpretations have been considered. Usually, a norm operator of one sort or another is incorporated into the optimisation which constrains the magnitudes of the values of $\boldsymbol{\beta}$, as seen in the well-studied ridge regression and lasso methods [157].

$$\boldsymbol{\beta}^{\text{ridge}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \boldsymbol{\beta}^T X\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

$$\boldsymbol{\beta}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \boldsymbol{\beta}^T X\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$$

where λ is a coefficient used to trade off between the regularisation and the data fit.

As one might imagine, the form of regulariser ultimately determines the effectiveness of the regression. Lasso, which uses the 1-norm has the characteristic of feature selection. This effect is caused because reductions in the magnitude of any β_i reduce the norm the same irrespective of the magnitude β_i, leading to more coefficients falling to zero. This allows the model to drop variables which is it does not believe effect prediction. A downside of lasso if that smoothly varying λ does not lead to smooth changes in the elements of $\boldsymbol{\beta}$. Ridge regression uses the 2-norm which penalises larger values of β_i far greater than smaller ones. As a result feature selection is not naturally possible, in the sense that feature weight would be exactly equal to zero, as all $\boldsymbol{\beta}$ add a negligible amount to the loss of the model but smooth changes in λ do lead to small changes in $\boldsymbol{\beta}$.

A strong advantage of frequentist regression is also our ability to choose a loss function. While the L_2-loss function nicely parallels the log-likelihood of a Gaussian distribution and L_1-loss function nicely parallels the log-likelihood of the ex-

ponential distribution, more complex loss functions do not have a clear analytic probabilistic interpretation. The soft margin loss function, made popular through the support vector machines and support vector regression, creates a margin of fixed width about the regressor in which data points are not penalised. Outside of this margin, data points then begin to accumulate loss. The standard hinge loss function, for example, grows linearly outside of this margin,

$$L^{\text{hinge}} = \sum_i \max\left\{|y_i - \boldsymbol{\beta}^T \mathbf{x}_i| - \epsilon, 0\right\}$$

There are many variants of regularisers and loss functions used throughout statistics and machine learning. Tradeoffs can be made in deciding upon a regulariser and even methods such as elastic nets which allow you to literally mix together lasso and ridge regression. For a full discussion on frequentist loss functions and regularisation I would strongly suggest referring to [61].

2.1.5 Classification

Classification is the task of learning a mapping between data and a classification or category. While classification is often referred to as a separate learning task to regression, at a fundamental level it is very much the same. Here we will focus on *logistic regression*, while support vector machines, decision trees and neural networks are discussed in the following chapters. Logistic regression is the classification generalisation of linear regression. Assuming binary output, $y \in \{0,1\}$, linear regression isn't really that useful as either $f(x)$ is either a constant for all $x \in [-\infty, \infty]$ or $f(x)$ is predicted to tend to $\pm\infty$ as x tends to $\pm\infty$. Instead, it would be desirable if $f(x)$ ranged between the bounds of the y values. This can be achieved by using a non-linear function of the linear model,

$$h_\beta(X) = g(\boldsymbol{\beta}^T X)$$

with,

$$g(z) = \frac{1}{1 + e^{-z}}$$

The function $g(x)$ is called the logistic function or sigmoid function, hence giving rise to the term logistic regression. By tuning $\boldsymbol{\beta}$ we are regressing the logistic function to fit the binary values. Due to the relationship between the logistic function and the concept of odds, $h_\beta(X)$ is often considered as the probability of the output being in the binary class of ones. This naturally gives rise to using MLE or other probability approaches in the optimisation of $\boldsymbol{\beta}$.

Often predictions of machine learning systems lead to a decision being made such as, for example, should a patient receive further treatment or should a load be approved. There are a number of additional considerations to make when performing a classification or more generally when making a decision. In practice, making incorrect classifications can have varying consequences depending on what the decision made was. In the binary decision scenario we decisions can be grouped into true-positives, true-negatives, false positives and false-negatives. The former two are when we correctly make a decision and the later two are when our model makes a mistake. In the case of a cancer screening system, for example, there may be a financial cost in performing more tests to someone incorrectly classified as having a malignant tumour. However, the cost of incorrectly classifying them as being cancer free may have serious consequences on their future health or even their chances of survival. There are a number of ways we can deal with these issues. We can create a confusion matrix and to choose an appropriate perfor-

mance metric which we wish to optimise for such as specificity, F1-score, precision and so on. The optimal decision of output class for logistic regression is when the output function is at 0.5, however, this threshold can be increased or decreased in order to optimise for the criteria which we wish to perform well against.

The general workflow for building a system in practice is to first develop a rich understanding for the context, that is how the underlying system works and the data available. Following this, we diagnose areas of the system which are creating bottlenecks or inefficiencies. We then define our evaluative measure, that is our criteria for success or metric which we wish to measure success by including constraints which may include interpretability, fairness, understanding of limitations amongst others. This in turn gives us our performance metric which we wish to optimise for. Finally, we propose methods and evaluate their performance based on the evaluative metric while taking into consideration experimental design, multiple comparison bias and other important experimental and statistical principals.

Logistic regression is a hugely important part of modern machine learning due to its profound impact in deep learning models and optimal decision making. This will be discussed in more detail in the next chapter.

2.2 Conclusion

This chapter has endeavoured to set out, at a high level, the concept of fitting lines to data via a range of approaches. The reason this is so important is that it underpins so much of machine learning, in particular regression. In particular, I aimed to outline the similarities between approaches, rather than their differences. This theme will be carried into the next chapter where we examine kernel methods.

Chapter 3

The Kernel Trick

In the previous chapter we have seen a brief overview of linear models which can be used for regression and classification. However, it comes as no surprise that linear models are inherently restrictive as it is trivial to find examples where a linear fit of the data, or split of the data, is unable to model the underlying function which has generated the data. This problem has been around for some time. In this chapter, we will outline design matrices and discuss the kernel trick.

3.1 Design Matrices

Design matrices offer a natural extension to linear models. The design matrix augments the input X with additional non-linear functions of X, referred to as explanatory variables. This is best shown via a simple example. Suppose X is a one dimensional input variable, but our belief is that the system, $X \xrightarrow{f(\cdot)} Y$, is polynomial in X of order d,

$$y = \sum_{i=0}^{d} x^i w_i,$$

then y is in fact a linear combination of X raised to each power from 0 through d. This linearisation of the problem allows us to express the non-linear regression in

a form akin to the linear models presented in the previous chapter,

$$\begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \dots & x_1^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n^1 & \dots & x_n^d \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_d \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

with $\epsilon_{i\,i=1}^{n}$ denoting the noise term. As this is clearly an additive extension of linear regression, it is referred to as additive modelling.

There are clear pros and cons when it comes to using design matrices for machine learning and statistics. On one hand, the model performs inference in a constrained manner using only easily interpretable functions of the input. This is important for a range of topics in statistics and economics where by the practitioner wishes to hypothesise and understand the dynamics of the system. Generalised additive models (GAMs) [82] have been used extensibly for explainable AI, where by the goal is not only to model the data and perform predictions, but also to be clearly interpretable by humans. These models are simply the linear combination of easily understandable functions of the input. On the other hand, by restricting the model to that of feature sets, hand-designed by the practitioner, the model may be severely limited. In practice many problems in machine learning have poorly interpretable feature sets. While linear trends maybe known a priori, the residual signal is not merely noise but perhaps a continuous stationary signal, for example. Kernels help us to better model such signals.

3.2 The Kernel Trick

The kernel trick, as it is commonly referred, extends the use of design matrices. The main premise is to replace the inner products XX^T by $\phi(X)\phi(X)^T$, with $\phi(\cdot)$ being some (potentially infinite dimensional) non-linear mapping from X to a feature

vector, with an analytic closed form equation $K(X,X) = \phi(X)\phi(X)^T$. This is best explained with an example derivation.

Suppose we have data in $[-1,1]$ and an infinite dimensional feature vector such that $\phi_i(x) = \exp(-x^2)\frac{\sqrt{2}x^i}{\sqrt{i!}}$. This feature space is referred to as the reproducing kernel Hilbert space (RKHS). Intuitively this would be impossible to write out explicitly. However, if we write out the inner product as a summation we find,

$$K(x_a, x_b) = \sum_{i=0}^{\infty} \exp(-x_a^2)\frac{\sqrt{2}x_a^i}{\sqrt{i!}} \exp(-x_b^2)\frac{\sqrt{2}x_b^i}{\sqrt{i!}}$$

Cleaning this summation up and noting the exponentials (independent of i) outside the summation and identifying that the series is the Taylor series of $\exp(2x_a x_b)$, we find that $K(x_a, x_b)$ has an elegant closed form,

$$K(x_a, x_b) = \exp(-(x_a - x_b)^2).$$

Kernels that do not live in an infinite dimensional RKHS are referred to as degenerate kernels and still have merit in many use cases.

3.2.1 Common Kernels

In the previous chapter, it was mentioned that under complete data fit requires either priors on the weights of the features or a loss function to penalise complex functions. In both cases likelihood and complexity are stated with respect to the weights of the dimensions of the RKHS. Thus in choosing a kernel function we are explicitly stating our beliefs on the types of functions we believe should be generating the data. Practitioners who use kernel methods thus need to know some of the common kernels available which they combine in order to encode their beliefs about the system. Below are a number of common kernels:

Linear Kernels

$$K(x_a, x_b) = \sigma_0^2 + \sigma_1^2(x_a - c)(x_b - c)$$

Polynomial Kernels

$$K(x_a, x_b) = (\sigma_0^2 + \sigma_1^2(x_a - c)(x_b - c))^d$$

Gaussian Kernels

$$K(x_a, x_b) = \sigma_0^2 \exp(-(x_a - x_b)^2/L)$$

Exponential Kernels

$$K(x_a, x_b) = \sigma_0^2 \exp(-|x_a - x_b|/L)$$

Hyperbolic Tangent Kernels

$$K(x_a, x_b) = \tanh(\sigma_0^2 + \sigma_1^2(x_a - c)(x_b - c))$$

Rational Quadratic Kernels

$$K(x_a, x_b) = \sigma_0^2 \left(1 + \frac{(x_a - x_b)^2}{l}\right)^{-\alpha}$$

Multiquadric Kernels

$$K(x_a, x_b) = \sigma_0^2 \sqrt{(x_a - x_b)^2 + c}$$

Inverse Multiquadric Kernels

$$K(x_a, x_b) = \sigma_0^2 \frac{1}{\sqrt{(x_a - x_b)^2 + c}}$$

Power Kernels

$$K(x_a, x_b) = -|x_a - x_b|^d$$

Log Kernels

$$K(x_a, x_b) = -\log(|x_a - x_b|^d + 1)$$

Cauchy Kernels

$$K(x_a, x_b) = \frac{1}{|x_a - x_b|^1 + 1}$$

Generalised Student-T Kernels

$$K(x_a, x_b) = \frac{1}{|x_a - x_b|^d + 1}$$

Periodic Kernels

$$K(x_a, x_b) = \sigma_0^2 \exp(-\sin((x_a - x_b)^2 / p) / L)$$

3.2.2 Kernel Combination

The beauty of kernel methods is that we can easily combine kernels to make more complex ones. This can easily be seen through our derivation at the beginning of this section when we examined the infinite dimensional RKHS and how the kernel function is simply the inner product in this RKHS. If one was to then ask if the addition of two functions was still a kernel this would simply be the equivalent of having an RKHS which appended the features from the first kernel to the second. Equally the product of two kernel functions would also be a kernel as the new RKHS would simply be the tensor product of the two feature spaces. Thus it is very easy to construct new kernels from old ones and this is exactly the principles set out in the Automated Statistician for kernel search [112].

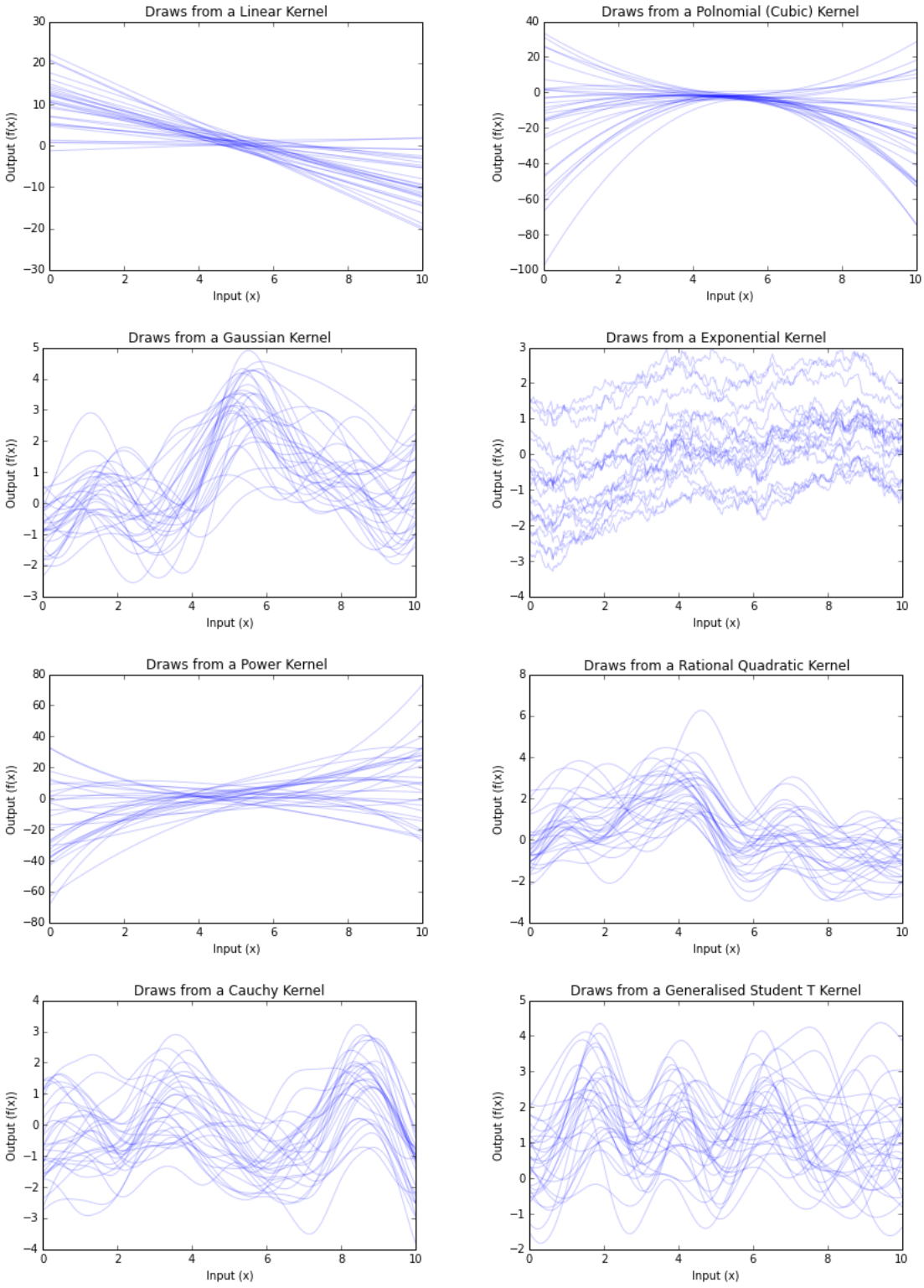


Figure 3.1: A selection of draws from Gaussian processes with various kernel priors.

3.3 Kernel Methods in Machine Learning

3.3.1 Gaussian Processes

Gaussian processes are Bayesian machine learning models which place Gaussian priors on both the feature space of the kernel and the observations. They are essentially an infinite dimensional generalisation of the Gaussian distribution where by each point in X maps to a unique dimension of the distribution. The covariance structure of the Gaussian is defined by the kernel function or the dot product in the feature Hilbert space. Preferable kernels which can model the space of all analytic functions have infinite dimensional reproducing kernel Hilbert spaces (RKHS) and there is rich theory which defines this more formally.

As such, Gaussian processes (GPs) place a probability distribution of the space of all function using a Gaussian prior. The conditional mean and covariance of the GP are as follow,

$$\mathbb{E}[f(x^*)] = K(x^*, x)K(x, x)^{-1}y$$

$$\mathbb{V}[f(x^*)] = K(x^*, x^*) - K(x^*, x)K(x, x)^{-1}K(x, x^*)$$

The propagation of both mean and uncertainty through a Gaussian process allows for uncertainty in the estimates made by the model. This has proven to be extremely useful in areas such as Bayesian optimisation [30], a global optimisation approach which exploits/exploits the space to find the expected optima according an acquisition function, for example the mean of the Gaussian process often encourages exploitation and the acquisition function explores the space by sampling where uncertainty is large.

Another important characteristic of a Gaussian process is that the definite integrand and derivative of a GP is also a GP provided the kernel is integrable and differentiable respectively [118, 119]. This can be easily shown by considering the Hilbert space view of the GP. As,

$$\begin{aligned} f(x) &= \sum_{i=0}^{\infty} w_i \phi_i(x), \\ \int_{\mathcal{R}^d} f(x)dx &= \int_{\mathcal{R}^d} \sum_{i=0}^{\infty} w_i \phi_i(x)dx \\ &= \sum_{i=0}^{\infty} \int_{\mathcal{R}^d} w_i \phi_i(x)dx \end{aligned} \tag{3.1}$$

The above integral operator can be replaced with the with differentiation or any other linear operator. Returning to the previous definition of the kernel function we find,

$$\begin{aligned} \begin{bmatrix} \int \phi(\bar{x})d\bar{x} \\ \phi(\bar{x}) \\ \frac{d\phi(\bar{x})}{dx} \end{bmatrix} \begin{bmatrix} \int \phi(x)dx \\ \phi(x) \\ \frac{d\phi(x)}{dx} \end{bmatrix}^T &= \begin{bmatrix} \int \phi(\bar{x})d\bar{x} \int \phi(x)dx & \phi(\bar{x}) \int \phi(x)dx & \frac{d\phi(\bar{x})}{d\bar{x}} \int \phi(x)dx \\ \int \phi(\bar{x})d\bar{x}\phi(x) & \phi(\bar{x})\phi(x) & \frac{d\phi(\bar{x})}{d\bar{x}}\phi(x) \\ \int \phi(\bar{x})d\bar{x}\frac{d\phi(x)}{dx} & \phi(\bar{x})\frac{d\phi(x)}{dx} & \frac{d\phi(\bar{x})}{d\bar{x}}\frac{d\phi(x)}{dx} \end{bmatrix} \\ &= \begin{bmatrix} \int \int K(\bar{x},x)d\bar{x}dx & \int K(\bar{x},x)dx & \int \frac{dK(\bar{x},x)}{d\bar{x}}dx \\ \int K(\bar{x},x)d\bar{x} & K(\bar{x},x) & \frac{dK(\bar{x},x)}{d\bar{x}} \\ \int \frac{K(\bar{x},x)}{dx}d\bar{x} & \frac{dK(x)}{dx} & \frac{dK(\bar{x},x)}{d\bar{x}dx} \end{bmatrix} \end{aligned} \tag{3.2}$$

This is an interesting result as it means quadrature and gradient observations of the function space can be incorporated into the GP just like any other observation. In a later chapter we will use this principle to create a general framework for fair regression.

3.3.2 Support Vector Machines

Support vector machines (SVMs) can be seen as the frequentist equivalent or extension of the Gaussian process. In practice they were more commonly used for

classification than Gaussian processes but have tailed off in popularity over recent years. Today neural networks and deep learning have somewhat overshadowed the SVM community. Support vector classification and regression use frequentist loss functions such as the hinge loss, described in the previous chapter, in order to fit discrimination boundaries or to regress function spaces. More details on SVMs can be found in [158].

3.3.3 Gauss Markov Random Fields

Gauss Markov Random Fields (GMRFs) [134] are very similar to Gaussian processes or more generally Gaussian distributions. Assume there is a field of nodes such that each node emits a Gaussian distribution. For the field to be a GMRF, each node must be independent of the rest of the field given a set of neighbouring nodes. More precisely, the precision matrix is sparse the non-zero values only between adjacent nodes in the field.

3.3.4 Determinental Point Processes

Determinental point processes (DPPs) [102] are stochastic point processes often used as dispersion models. The conditional probability of a set of points is equal to the determinant of the kernel matrix,

$$P(X) = |K(X, X)|.$$

DPPs are often used in practice for news recommendations whereby the algorithm designer would like to present a random selection of news items with low probability of similar items appearing together. They also have a nice theoretical foundation as modelling the zero crossings of Gaussian processes [84].

3.4 Links to Other Machine Learning Models

While kernel methods are usually seen as a distinct set of models in machine learning, they are in fact very much related to a broader range of models. In this section we will outline some of the simple similarities between kernel methods and other popular machine learning models.

3.4.1 Decision Trees and Random Forests

Decision trees are models which recursively subdivide the input space using orthogonal partitions. The output for a particular decision is voted by the data in the leaf of the tree, in the case of regression, averaged over the elements in the leaf. They are often one of the first machine learning approaches taught to students as they are very intuitive and are highly interpretable as they can be explained through a number of conditional if statements over the input variables. Random forests, in their most simple of description, are a combination of many decision trees each of which vote and whose decisions are then combined.

We can encode a decision tree easily into a kernel function as,

$$K(x_i, x_j) = \begin{cases} \alpha, & \text{if } l(x_i) = l(x_j) \\ 0, & \text{otherwise} \end{cases}$$

for some $\alpha > 0$ where $l(\cdot)$ indexes the leaf of the tree which the data is mapped to. The expectation of this process, using pseudo inversion as the kernel is naturally degenerate having the number of eigenvalues equal to the number of leaves, returns the same prediction as the decision tree.

Interestingly, if the noise is Gaussian and constant across each Gaussian process 'tree' in a Bayesian committee machine then the combined mean function is exactly equivalent to tree bagging. Further, if each GP only uses a random subset

of input dimensions, the output of the mean function is exactly equivalent to that of a random forest.

3.4.2 Neural Networks

Neural networks and deep neural networks have displayed huge advances in the applied domains of machine learning over recent years. State-of-the-art performance in face recognition, image recognition, voice to text, automatic translation, image segmentation and many more domains has been transformed by their application. That said, neural networks are combinations of relatively simple primitives. The single neuron is simply a function of the form $f(WX + b)$, where $f(\cdot)$ is some non-linear function. The goal of a neural network for classification or regression is to learn the weight matrices W and bias vectors b such that in the transformed space simple linear regression is highly accurate. Unfortunately, standard gradient descent is not always effective and various optimisation techniques have been considered [96]. Similarly, for images, audio and other mediums it has been shown to be more effective to use convolutional neurons which act like filters on the input rather than static weights on fixed regions.

Stepping back from the nuances of the architecture of the neural networks we can consider them as learning a finite dimensional feature map $\phi(x)$ (the output logits are the explicit feature space), with which degenerate kernel learning may be achieved. Gaussian processes and SVMs tend to use kernels which have intuitive properties and are generally speaking non-degenerate. Priors are often placed on their hyper-parameters, either explicitly in the case of GPs or implicitly in the case of the loss functions of the SVM. Neural networks, in contrast, have implicit priors based on the architecture of the network. They are degenerate and thus cannot

model every function, but due to their high efficiency they generally allow for many parameters and often mimic non-degenerate kernels in practice.

Importantly note that classification and regression are ultimately achieved through traditional means and hence general statements can still be made. An example of this will be in the next chapter whereby a general framework for fair regression is presented.

3.5 Conclusion

In this chapter, I have introduced generalised linear regression via the design matrices and later shown how these can naturally be extended to kernel methods, as was shown via the expansion of the Gaussian kernel. While not always emphasised, kernel methods dominate machine learning literature in classification and regression, from Gaussian processes, support vector machines, decision trees, neural networks (multi-layer perceptrons, convolutional neural networks, recurrent neural networks), determinental point processes, Gauss Markov random fields and more. This is a truly important take-home which I believes has allowed me to translate insights from one sub-domain to another, with applications to transfer learning, constrained inference and more. One such example of this is outlined in the next chapter.

Chapter 4

Constrained Kernel Regression with Applications to Group Fairness

Fairness, through its many forms and definitions, has become an important issue facing the machine learning community. In this work, we consider how to incorporate group fairness constraints in kernel regression methods, applicable to Gaussian processes, support vector machines, neural network regression and decision tree regression. Further, we focus on examining the effect of incorporating these constraints in decision tree regression, with direct applications to random forests and boosted trees amongst other widespread popular inference techniques. We show that the order of complexity of memory and computation is preserved for such models and tightly bound the expected perturbations to the model in terms of the number of leaves of the trees. Importantly, the approach works on trained models and hence can be easily applied to models in current use and group labels are only required on training data.

As the proliferation of machine learning and algorithmic decision making continues to grow throughout industry, the net societal impact of them has been studied with more scrutiny. In the USA under the Obama administration, a report on big data collection and analysis found that "big data technologies can cause soci-

etal harms beyond damages to privacy" [155]. The report feared that algorithmic decisions informed by big data may have harmful biases, further discriminating against disadvantaged groups. This along with other similar findings has led to a surge in research around algorithmic fairness and the removal of bias from big data.

The term *fairness*, with respect to some sensitive feature or set of features, has a range of potential definitions. In this work, impact parity is considered. In particular, this work is concerned with group fairness under the following definitions as taken from [62],

Group Fairness: *A predictor $\mathcal{H} : X \to Y$ achieves fairness with bias ϵ with respect to groups $A, B \subseteq X$ and $O \subseteq Y$ being any subset of outcomes iff,*

$$|\mathrm{P}\{\mathcal{H}(x_i) \in O | x_i \in A\} - \mathrm{P}\{\mathcal{H}(x_j) \in O | x_j \in B\}| \leq \epsilon$$

The above definition can also be described as statistical or demographical parity. Group fairness has found widespread application in India and the USA, where affirmative action has been used to address discrimination against caste, race and gender [161, 48, 44].

The above definition does not, unfortunately, have natural application to regression problems. One approach to get around this would be to alter the definition to bound the absolute difference between the respective marginal distributions over the output space. However, this is a strong requirement and may hinder the model's ability to model the function space appropriately. Rather, a weaker and potentially more desirable constraint would be to force the expectation of the marginal distributions over the output space to equate. Therefore, statements such as *"the average expected outcome for population A and B is equal"* would be valid.

The second issue encountered is that the generative distribution of groups A

and B are generally unknown. In this work, it is assumed that the empirical distribution $p_A(x)$ and $p_B(x)$, as observed from the training set, is equal to or negligibly perturbed from the true generative distributions.

Combining these two caveats we arrive at the following definition,

Group Fairness in Expectation: *A regressor $f(\cdot) : X \to Y$ achieves fairness with respect to groups $A, B \subseteq X$ iff,*

$$\mathbb{E}[f(x_i)|x_i \in A] - \mathbb{E}[f(x_i)|x_j \in B] = 0$$

$$\int (p_A(x) - p_B(x))\, f(x) dx = 0$$

There are many machine learning techniques with which *Group Fairness in Expectation* constraints (GFE constraints) may be incorporated. While constraining kernel regression is introduced in Section 3, the main focus of the chapter is examining decision tree regression and respective ensemble methods which build on decision tree regression such as random forests, extra trees and boosted trees due to their widespread use in industry and hence their extensive impact on society [166]. The reason for this is to show that such an approach will not affect the order of computational or memory complexity of the model.

The main contributions of this chapter are,

I The use of quadrature approaches to enforce GFE constraints on kernel regression with applications to Gaussian processes, support vector machines, neural network regression and decision tree regression, as outlined in Section 3.

II Incorporating these constraints on decision tree regression without affecting the computational or memory requirements, outlined in Sections 5 and 6.

III Deriving a tight bound for the variance of the perturbations due to the incorporation of GFE constraints on decision tree regression in terms of the number of leaves of the tree, outlined in Section 7.

IV Showing that these *fair* trees can be combined into random forests, boosted trees and other ensemble approaches while maintaining fairness, as shown in Section 8.

4.1 Related Work

There are many ways in which the now huge volume of literature on algorithmic fairness may be split. One such approach is to break the proposed literature into three branches of research based upon the stage of the machine learning life cycle they belong. The first is the data alteration approach, which endeavours to modify the original dataset in order to prevent discrimination or bias due to the protected variable [105, 91]. The second is an attempt to regularise such that the model is penalised for bias [93, 19, 33, 34, 127]. Finally, the third endeavours to use post-processing to re-calibrate and mitigate against bias [92, 124].

The literature also differs dramatically as to what is the objective of the fairness algorithm. Recent work has made efforts towards grouping these into consistent objective formalisation [62, 99]. Often, the focus of algorithmic fairness is on classification problem with regression receiving very little attention.

The approach applied to enforce fairness may be from a plethora of definitions, anti-classification [99], or fairness through unawareness as it is also referred to as [62], endeavour to treat data agnostic of protected variables and hence enforces fairness via treatment rather than outcome. The second popular method is classification parity, i.e., the error with respect to some given measure is equal across

groups defined by the protected variable. Finally, calibration is the term used when outcomes are independent of protected group conditioned on risk.

Narrowing our focus to regression, two contradicting objectives once again arise, namely group level fairness and individual fairness. Individual fairness implies that small changes to a given characteristic of an individual leads to small changes in outcome. Group fairness on the other hand endeavours to make aggregate outcomes of protected groups similar. The latter is the focus of this work and an overview of where this fits into the broader litterature may be found in Table 4.1.

Table 4.1: This table is amended from [62], highlighting some of the major contribution currently in the domain of fairness in machine learning. Parity versus preference refers to whether fairness means achieving equality or satisfying the preferences. Treatment versus impact refers to whether fairness is to be maintained in treatment or process of the learning algorithm or resulting output of the system. To the best of the authors knowledge, this work is the first group fair framework for regression problems.

	Parity		**Preference**
Treatment	Unawareness [50, 105]	Counterfactual [95, 101, 94]	Preferred Treatment [168]
Outcome	Group Fairness [52, 46, 128]	Individual Fairness [19, 50, 105]	Preferred Impact [168, 13]
	Equal Opportunity [90, 18, 80, 167]		

Our work endeavours to create group level parity of expected outcome, of Group Fairness in Expectation as introduced in this work, with application to all kernel based regression which minimise the L2 norm. This includes decision trees, Gaussian processes and multi-layer perceptrons.

Specifically to decision trees, discrimination aware decision trees have been introduced [**?**] for classification. They offer dependency aware tree construction and

leaf relabelling approach. Later, fair forests [127] introduced a further tree induction algorithm to encourage fairness. They did this by introducing a new gain measure to encourage fairness. However, the issue with adding such regularisation is two-fold. Firstly, discouraging bias via a regularising term does not make any guarantee about the bias of the post trained model. Secondly, it is hard to make any theoretical guarantees about the underlying model or the effect the new regulariser has had on the model.

The approach offered in this work seeks to perform model inference in a constrained space, leveraging basic theory from Bayesian quadrature such that the predicted marginal distributions are guaranteed to have equal means. Such moment constraints have a natural relationship to maximum entropy methods. By utilising quadrature methods, it is also possible to derive bounds for the expected absolute perturbation induced by constraining the space. This is shown explicitly in the following sections. Ultimately, the paper develops a general framework to perform group-fair regression, an important open problem as pointed out in [46].

We emphasise to the reader that, as outlined in the next section, there are many definitions of fairness, each with reasonable motives but conflicting values. Group fairness, addressed in this work, inherently leads to individual unfairness, i.e., to create equal aggregate statistics between sub-population, individuals in each sub-population are treated inconsistently. The reverse is also true. As such, we should always think through the adverse effects of our approach before applying it in the real world. The experiments in this paper are aimed to explore and demonstrate the approach introduced, but are not meant to advocate using group fairness specifically for the task in hand.

4.2 Constrained Kernel Regression

We will first show how one can create such linear constraints on kernel regression models. This work builds on the earlier contributions of [88], where the authors examined the incorporation of linear constraints on Gaussian processes (GPs). Gaussian processes are a Bayesian kernel method most popular for regression. For a detailed introduction to Gaussian processes, we refer the reader to [131]. However, for the reader unfamiliar with GPs specifically, they may simply think of a high dimensional Gaussian distribution parameterized by a kernel $K(\cdot,\cdot)$, with zero mean and unit variance without loss of generality. Given a set of inputs and respective outputs, $\{\hat{x}_i, \hat{y}_i\}_{i=1}^{N}$, split into training and testing sets,$\{\mathrm{x}_i, \mathrm{y}_i\}_{i=1}^{n}$ and $\{\bar{\mathrm{x}}_i, \bar{\mathrm{y}}_i\}_{i=1}^{N-n}$, inference is performed as,

$$\mathbb{E}[\bar{\mathrm{y}}] = K_{\bar{x},x} K_{x,x}^{-1} \mathrm{y}$$

$$\mathbb{V}[\bar{\mathrm{y}}] = K_{\bar{x},\bar{x}} - K_{\bar{x},x} K_{x,x}^{-1} K_{x,\bar{x}}$$

where $K_{x,x}$ denotes the kernel matrix between training examples, $K_{\bar{x},x}$ is the kernel matrix between the test and training examples and $K_{\bar{x},\bar{x}}$ is the prior variance on the prediction point defined by the kernel matrix. Gaussian processes differ from high dimensional Gaussian distributions as they can model the relationships between points in continuous space, via the kernel function, as opposed to being limited to a finite dimension.

An important note is that any combination of Gaussian distributions via addition and subtraction is a closed space, that is to say, the sum of Gaussians is also Gaussian and so on. While this may at first appear trivial, it is, in fact, a very useful artefact. For example, let us assume there are two variables, a and b, drawn from

Gaussian distributions with mean and variance $\mu_a, \mu_b, \sigma_a^2, \sigma_b^2$ respectively. Further, assume that the correlation coefficient ρ describes the interaction between the two variables. Then a new variable c, which is equal to the difference a and b, is drawn from a Gaussian distribution with mean and variance,

$$\mu_c = \mu_a - \mu_b$$

$$\sigma_c^2 = \sigma_a^2 + \sigma_b^2 - 2\rho\sigma_a\sigma_b$$

We can thus write all three variables in terms of a single mean vector and covariance matrix,

$$\mu = \begin{bmatrix} \mu_a - \mu_b \\ \mu_a \\ \mu_b \end{bmatrix}$$

$$K = \begin{bmatrix} \sigma_a^2 + \sigma_b^2 - 2\rho\sigma_a\sigma_b & \sigma_a^2 - \rho\sigma_a\sigma_b & \sigma_b^2 - \rho\sigma_a\sigma_b \\ \sigma_a^2 - \rho\sigma_a\sigma_b & \sigma_a^2 & \rho\sigma_a\sigma_b \\ \sigma_b^2 - \rho\sigma_a\sigma_b & \rho\sigma_a\sigma_b & \sigma_b^2 \end{bmatrix}$$

Given observations of any two of the dimensions above, the third can be inferred exactly. We refer to this as a degenerate distribution as K will naturally be low rank. If we impose that $\mu_a - \mu_b$ is equal to zero, we are thus constraining the distribution of a and b. This can easily be extended to the relationship between sums and differences of more variables.

Bayesian quadrature [118] is a technique used to incorporate integral observations into the Gaussian process framework. Essentially, quadrature can be derived through an infinite summation and the above relationship between these summations can be exploited [119]. An example covariance structure thus looks akin to,

$$K = \begin{bmatrix} \int\int p(x)K(x,x')p(x')\mathrm{d}x\mathrm{d}x' & \int p(x)K(x,x_0)\mathrm{d}x & \int p(x)K(x,x_1)\mathrm{d}x \\ \int p(x)K(x,x_0)\mathrm{d}x & K(x_0,x_0) & K(x_1,x_0) \\ \int p(x)K(x,x_1)\mathrm{d}x & K(x_1,x_0) & K(x_1,x_1) \end{bmatrix}$$

where $p(x)$ is some probability distribution over the domain of x, on which the Gaussian process is defined and against which the quadrature is performed against.

Reiterating the the motivation of this work; given two generative distributions $p_A(x)$ and $p_B(x)$ which subpopulations A and B of the data are generated from, we wish to constrain the inferred function $f(\cdot)$ such that,

$$\int p_A(x)f(x)\mathrm{d}x = \int p_B(x)f(x)\mathrm{d}x.$$

This constraint can be rewritten as,

$$\int \left(p_A(x) - p_B(x)\right) f(x)\mathrm{d}x = 0,$$

which allows us to incorporate the constraint on $f(\cdot)$ as an observation in the above Gaussian process. Let $q_{A,B}(x) = p_A(x) - p_B(x)$ be the difference between the generative probability distributions of A and B, then by setting the corresponding observation as zero, the covariance matrix becomes,

$$K = \begin{bmatrix} \int\int q_{A,B}(x)K(x,x')q_{A,B}(x')\mathrm{d}x\mathrm{d}x' & \int q_{A,B}(x)K(x,x_0)\mathrm{d}x & \int q_{A,B}(x)K(x,x_1)\mathrm{d}x \\ \int q_{A,B}(x)K(x,x_0)\mathrm{d}x & K(x_0,x_0) & K(x_1,x_0) \\ \int q_{A,B}(x)K(x,x_1)\mathrm{d}x & K(x_1,x_0) & K(x_1,x_1) \end{bmatrix}$$

We will refer to these as equality constrained Gaussian processes. Let us now turn to incorporate these concepts into decision tree regression.

4.3 Trees As Kernel Regression

Decision tree regression (DTR) and related approaches offer a white box approach for practitioners who wish to use them. These methods are among the most popular methods in machine learning [166] in practice as they are generally intuitive even for those not from statistics, mathematics or computer science background. It is their proliferation, especially in businesses without machine learning

researchers, that makes them of particular interest.

DTR regress data by sorting them down binary trees based partitions in the input domain. The trees are created by recursively partitioning the domain of input along axis aligned splits determined by a given metric of the data in each partition, such as information gain or variance reduction. In this work, we will not consider the many possible techniques for learning decision trees, but rather assume that the practitioner has a trained decision tree model. For a more complete description of decision trees, the authors refer the readers to [133].

For the purposes of this work, DTR can be described as a partitioning of space such that predictions are made by averaging the observations in the local partition, referred to as the leaves of the tree. As such, DTR has a very natural formulation as a degenerate kernel whereby,

$$K(x_i, x_j) = \begin{cases} 1, & \text{if } L(x_i) = L(x_j) \\ 0, & \text{otherwise} \end{cases}$$

where $L(\cdot)$ is the index of the leaf in which the argument belongs. The kernel matrix becomes naturally block diagonal, as data on leaves are simply averaged, and the classifier / regressor written as,

$$f(\bar{\mathrm{x}}) = \mathbb{E}[\bar{\mathrm{y}}|\bar{\mathrm{x}}] = K_{\bar{x},x} K_{x,x}^{-1} \mathrm{y}$$

with $K_{\bar{x},x}$ denoting the vector of kernel values between $\bar{\mathrm{x}}$ and the observations, $K_{x,x}$ denotes the covariance matrix of the observations as defined by the implicit decision tree kernel and y denoting the values of the observations.

It is worth also noting how one can also write the decision tree as a two-stage model: first by averaging the observations of associated with each leaf and then by using a diagonal kernel matrix to perform inference. Trivially, as there are perfect

correlations between data on a each leaf, the diagonal kernel matrix acts only as a lookup and outputs the leaf average that corresponds to the point being predicted. Let us refer to this compressed kernel matrix approach as the *compressed kernel representation* and the block diagonal variant as the *explicit kernel representation*.

4.4 Fairness Constrained Decision Trees

Borrowing concepts from the previous section on equality constrained Gaussian processes using Bayesian quadrature, decision trees may be constrained in a similar fashion. The first consideration to note is that we wish the constraint observation to act as a hard equality, that is to say noiseless. In contrast, we are willing for the observations to be perturbed in order to satisfy this hard equality constraint. To achieve this, let us add a constant noise term, σ^2_{noise}, to the diagonals of the decision tree kernel matrix. Similar to ordinary least squares regression, the regressor will now minimize the L2-norm of the error induced on the observations, conditioned on the equality constraint which is noise free. In the explicit kernel representation, this implies the minimum induced noise *per observation*, whereas in compressed kernel representation this implies the minimum induced noise *per leaf*.

An important note is that the constraint is applied to the kernel regressor equations, hence the method is exact for regression trees or when the practitioner is concerned with relative outcomes of various predictions. However, in the case that the observations range between $[0,1]$, as is the case in classification, then we must renormalize the output to $[0,1]$. This no longer guarantees a minimum L2-norm perturbation and while potentially still useful, is not the focus of this work.

The second consideration is how to determine the generative probability distributions $p_A(x)$ and $p_B(x)$. Given the frequentist nature of decision trees, it makes

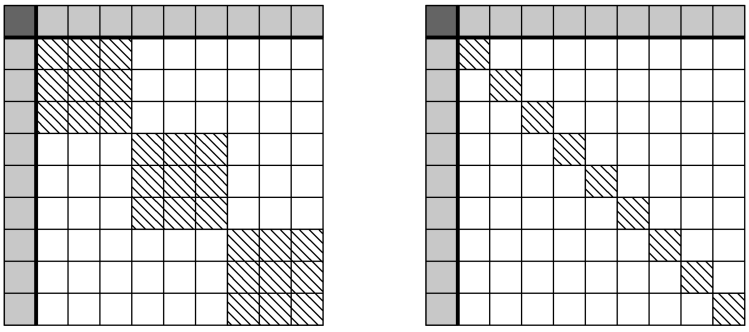


Figure 4.1: Above is a visualization of a decision tree kernel matrix with marginal constraint, left in explicit representation and right in compressed representation. The dark cell in the upper left of the matrix is the double integrated kernel function with respect to the difference of input distributions which constrain the process. The solid grey row and column are single integrals of the kernel function. White cells have zero values and the dashed (block) diagonals are the kernel matrix between observations or leaves of the tree. We can note that the above, compressed representation kernel matrix is an arrowhead matrix, which we will exploit to create an efficient algorithm.

sense to consider $p_A(x)$ and $p_B(x)$ as the empirical distributions of subpopulations A and B as described in Section 1. Thus the integral of the empirical distribution on a given leaf, $\int_{L_i} p_A(x)\mathrm{d}x$, is defined as the proportion of population A observed in the partition associated with leaf L_i. We emphasise that how $p_A(x)$ and $p_B(x)$ are determined is not the core focus of this work and many approaches have merit. For example, a Gaussian mixture model could be used to model the input distribution in which case $\int_{L_i} p_A(x)\mathrm{d}x$ would equal the cumulative distribution of the generative PDF over the bounds defined by the leaf. This is demonstrated in the experimental section. Many other such models would also be valid and determining which method to use to model the generative distribution is left to the practitioner with domain expertise.

4.5 Efficient Algorithm For Equality Constrained Decision Trees

At this point, an equality constrained variant of a decision tree has been described, both in explicit representation and compressed representation. In this section, we will show that equality constraints on a decision tree do not change the computational or memory order of complexity. The motivation for considering the order of complexities is that decision trees are one of the more scalable machine learning models, whereas kernel methods such as Gaussian processes naively scale at $\mathcal{O}(n^3)$ in computation and $\mathcal{O}(n^2)$ in memory, where n is the number of observations. While the approach presented in this work utilizes concepts from Bayesian quadrature and linearly constrained Gaussian processes, the model's usefulness would be drastically hindered if it no longer maintained the performance characteristics of the classic decision tree, namely computational cost, and memory requirements.

4.5.1 Efficiently Constrained Decision Trees in Compressed Kernel Representation

As Figure 1 shows, the compressed kernel representation of the constrained decision tree creates an arrowhead matrix. It is well known that the inverse of an arrowhead matrix is a diagonal matrix with a rank-1 update. Letting D represent the diagonal principal sub-matrix with diagonal elements equal to one, z being vector such that the i^{th} element is equal to the relative difference in generative populations distributions for leaf i, $z_i = \int_{L_i}(p_A(x) - p_B(x))dx$, then the arrowhead inversion properties state that,

$$\begin{bmatrix} D & z \\ z^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} D^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \rho u u^T,$$

with $\rho = -\frac{1}{z^T D^{-1} z}$ and $u = \begin{bmatrix} D^{-1} z \\ -1 \end{bmatrix}$. Note that the integral of the difference between the two generative distributions when evaluated over the entire domain is equal to zero, as both $p_A(x)$ and $p_B(x)$ must sum to one by definition and hence their differences to zero. Returning to the equation of interest, namely $f(\bar{\mathrm{x}}) = K_{\bar{x},x} K_{x,x}^{-1} \mathrm{y}$ with y as the average value of each leaf of the tree, and subbing in $K_{\bar{x},x}$ as a vector of zeros with a one indexing the j^{th} leaf in which the predicted point belongs to and is equal to zero, as it does not contribute to the empirical distributions, we arrive at,

$$f(\bar{\mathrm{x}}) = \frac{1}{1+\sigma_n^2} \left(\mathrm{y}_j + z_j \frac{\sum_i z_i y_i}{\sum_i z_i^2} \right).$$

The term $\frac{1}{1+\sigma_n^2}$ is the effect of the prior under the Gaussian process perspective, however, by post-multiplying by $(1+\sigma_n^2)$, this prior effect can be removed. While relatively simple to derive, the above equation shows that only an additive update to the predictions is required to ensure group fairness in decision trees. Further, if the same relative population is observed for group A and group B on a single leaf j, then $z_j = 0$ and no change is applied to the original inferred prediction before the constraint is applied other than the effect of the noise. In fact, the perturbation to a leaf's expectation grows linearly with the bias in the population of the leaf.

From an efficiency standpoint, only the difference in generative distributions, z, needs to be stored which is an additional $\mathcal{O}(L)$ extra memory requirement and the update per leaf can be pre-computed in $\mathcal{O}(L)$. These additional memory and computational requirements are negligible compared to $\mathcal{O}(N)$ cost of the decision tree itself.

4.5.2 Efficiently Constrained Decision Trees in Explicit Kernel Representation

Let us now turn our attention to the explicit kernel representation case, that is where the D in the previous subsection is replaced with the block diagonal matrix equivalent. First let us state the *bordering method*, a special case of the block diagonal inversion lemma,

$$\begin{bmatrix} D & z \\ z^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} D^{-1} - \rho D^{-1} z z^T D^{-1} & \rho D^{-1} z \\ \rho z^T D^{-1} & -\rho \end{bmatrix}.$$

with $\rho = -\frac{1}{z^T D^{-1} z}$ once again. Substituting this into the kernel regression equation once more we find,

$$\begin{aligned} f(\bar{\mathrm{x}}) &= \begin{bmatrix} \mathbb{I}_j \\ 0 \end{bmatrix} \begin{bmatrix} D & z \\ z^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{y} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbb{I}_j \\ 0 \end{bmatrix} \begin{bmatrix} D^{-1} - \rho D^{-1} z z^T D^{-1} & \rho D^{-1} z \\ \rho z^T D^{-1} & -\rho \end{bmatrix} \begin{bmatrix} \mathrm{y} \\ 0 \end{bmatrix} \end{aligned}$$

where $\mathbb{I}_j$ denotes a vector of zeros with ones placed in all elements relating to observations in the same leaf. Expanding the above linear algebra,

$$f(\bar{\mathrm{x}}) = \mathbb{I}_j D^{-1} \mathrm{y} + \rho \mathbb{I}_j D^{-1} z z^T D^{-1} \mathrm{y}.$$

As D is a block-diagonal matrix, it is straight forward to show $\rho = -\left(\sum_{j \in L} \frac{m_j z_j^2}{m_j + \sigma_n^2} \right)^{-1}$ where j is iterating over the set of leaves. Note that when $m_j = 1$ for all j we arrive at the same value for ρ as we did in the previous subsection. We can continue to apply this result to the other terms of interest,

$$\begin{aligned} X_1 &= \mathbb{I}_j D^{-1} \mathrm{y} = \frac{m_j \mathrm{y}_j}{m_j + \sigma_n^2} \\ X_2 &= \mathbb{I}_j D^{-1} z = \frac{m_j z_j}{m_j + \sigma_j^2} \\ X_3 &= z^T D^{-1} \mathrm{y} = \sum_{j \in L} \frac{m_j z_j \mathrm{y}_j}{m_j + \sigma_n^2} \end{aligned}$$

where y_j is once again the average output observation over leaf j. The terms have been labelled X_1, X_2 and X_3 for shorthand. The computation time for the three terms, along with ρ, can be computed in linear time with respect to the size of the data, $\mathcal{O}(n)$, and can be pre-computed ahead of time, hence not affect the computational complexity of a standard decision tree. Once again only z_j and m_j have to be stored for each leaf and hence the additional memory cost is only $\mathcal{O}(L)$. As such we can simplify the full expression for the expected outcome as,

$$f(\bar{\mathrm{x}}) = X_1 + \rho X_2 X_3.$$

4.6 Expected Perturbation Bounds

In imposing equality constraints on the models the inferred outputs become perturbed. In this section, the expected magnitude of the perturbation is analyzed for the compressed kernel representation. We define the perturbation due to the equality constraint, not due to the incorporation of the noise, as,

$$\epsilon = z_j \frac{\sum_i z_i y_i}{\sum_i z_i^2}.$$

Theorem 1 *Given a decision tree with L leaves, with expected value of leaf observations denoted by the vector $y \in \mathbb{R}^L$ normalized to have zero mean and unit variance and leaf frequency imbalance denoted as $z \in \mathbb{R}^L$ the expected variance induced by the perturbation due to the incorporating a Group Fairness in Expectation constraint is bounded by,*

$$\mathbb{E}[\epsilon^2] \leq \frac{1}{L}$$

Proof 1 As the expectation of z_j is zero due to it being the difference of two probability distributions, the variance is equal to the expectation of ϵ^2,

$$\mathbb{E}[\epsilon^2] = \frac{1}{L}\sum_{j=1}^{L} z_j^2 \frac{(\sum_i z_i y_i)^2}{(\sum_i z_i^2)^2} = \frac{1}{L}\|z\|_1^2 \frac{\|z\|_2^2 (\bar{z}^T y)^2}{\|z\|_2^4}$$

with $\bar{z}$ equal to z after normalization. By Lemma 1, the expectation of the dot product $(\bar{z}^T y)^2$ is equal to $\frac{1}{L}$. Further, the 2-norm of z can be cancelled from the numerator and denominator. Finally, using the L_1, L_2 norm inequality, $\|z\|_2 \leq \|z\|_1 \leq \sqrt{L}\|z\|_2$, we can then tightly bound the worst case introduced variance as,

$$\mathbb{E}[\epsilon^2] = \frac{1}{L^2}\|z\|_1^2 \|z\|_2^{-2} \leq \frac{1}{L}$$

Lemma 1 *Given two vectors $y, \bar{z}$ uniformly distributed on the unit hypersphere, $\mathcal{S}^{L-1}$ using n-sphere notation, the expectation of their dot product is zero and variance,*

$$\mathbb{E}[(\bar{z}^T y)^2] = \frac{1}{L}.$$

Proof 2 As the inner product is rotation invariant when applied to both $\bar{z}$ and y, let us denote the vector $\bar{z}$ as $[1, 0, \ldots, 0]$ without loss of generality. The first element of the vector y, denoted by y_0, will thus be equal to $\bar{z}^T y$. The probability density mass of the random variable y_0 is proportional to the surface area lying at a height between y_0 and $y_0 + dy_0$ on the unit hypersphere. That proportion occurs within a belt of height dy_0 and radius $\sqrt{1 - y_0^2}$, which is a conical frustum constructed out of an S^{L-2} of radius $\sqrt{1 - y_0^2}$, of height dy_0, and slope $\frac{1}{\sqrt{1-y_0^2}}$. Hence the probability is proportional to,

$$P(y_0) \sim \frac{(\sqrt{1 - y_0^2})^{D-2}}{\sqrt{1 - y_0^2}} dy_0 = (1 - y_0^2)^{\frac{L-3}{2}}$$

Substituting $u = \frac{y_0+1}{2}$ we find that,

$$
\begin{aligned}
P(u)du &\sim (1-(2u-1)^2)^{\frac{L-3}{2}} d(2u-1) \\
&= 2^{L-2}(u-u^2)^{\frac{L-3}{2}} du \\
&= 2^{L-2} u^{\frac{L-1}{2}-1}(u-1)^{\frac{L-1}{2}-1} du.
\end{aligned}
$$

Note that this last simplification of $P(u)$ is equal to the probability density function of the Beta distribution with both shape parameters equal $\alpha = \beta = \frac{L-1}{2}$. The variance of the Beta distribution is,

$$
\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} = \frac{1}{4L}.
$$

Rescaling to find the variance of y_0 we arrive at $\frac{1}{L}$. As the expectation of $\mathbb{E}[\bar{z}^T y] = 0$ due to the properties of symmetry, $\mathbb{E}[(\bar{z}^T y)^2] = \frac{1}{L}$.

This is an interesting result as it implies that if the model is not exploiting biases in the generative distribution evenly across all of the leaves of the tree, that is to say, $\|z\|_1 = \sqrt{L}\|z\|_2$, then the resulting predictions will receive the greatest expected absolute perturbation when averaged over all possible y.

For the explicit kernel representation, the expected absolute perturbation bound can be analysed whereby each leaf holds an even number of observations. In such a scenario $m_i = m$ is equal for all leaves $i \in 1, \ldots, L$. Substituting this into the equations for ρ, X_2 and X_3 we can find that the bounded expected perturbation is equal to,

$$
\mathbb{E}[\epsilon^2] \leq \left(\frac{\sigma_n^2}{m+\sigma_n^2}\right)^2 \frac{1}{L}
$$

For the sake of conciseness the full derivation of the above is left to the reader but follows the same steps as the compressed kernel representation.

4.7 Combinations Of Fair Trees

While it is intuitive to say that ensembles of trees with GFE constraints preserve the GFE constraint, however, for the sake of completeness this is now shown more formally. Random forests [29], extremely random trees (ExtraTrees) [66] and tree bagging models [28] combine tree models by averaging over their predictions. Denoting the predictions of the trees at point x as $f_i(x)$ for each $i \in 1, \ldots, T$, where T is the number of trees, we can easily show that the combined difference in expectation marginalised over the space is equal to zero,

$$\begin{aligned} 0 &= \int (p_A(x) - p_B(x)) \sum_{i=1}^{T} f_i(x) \mathrm{d}x \\ &= \sum_{i=1}^{T} \int (p_A(x) - p_B(x)) f_i(x) \mathrm{d}x = \sum_{i=1}^{T} 0 \end{aligned}$$

It can also be easily shown that modelling residual errors of the trees with other fair trees, such as is the case for boosted tree models [54], results in fair predictors also. These concepts are not limited to tree methods either and the core concepts set out in this shapter of constraining kernel matrices can have applications in models such as deep Gaussian process models [40].

4.8 Experiments

4.8.1 Synthetic Demonstration

The first experiment is a visual demonstration to better communicate the validity of the approach. The models examined are ExtraTrees, Gaussian processes and a single hidden layer perceptron. They endeavour to model an analytic function, $f(x) = x\cos(\alpha x^2) + \sin(\beta x)$, with observations drawn from two beta distributions, $p_A(x)$ and $p_B(x)$ respectively. The parameters of the two beta distribution are,

Group	$p_A(x)$	$p_B(x)$
α	2	3
β	3	2

Table 4.2: Parameters of Beta distributions used to create synthetic samples.

Figure 4.5 shows the effect of perturbing the models using the approach presented to constrain the expected means of the two populations. The figure shows the greater disparity between $p_A(x)$ and $p_B(x)$, the greater the perturbation in the inferred function. Both the compressed and explicit kernel representation lead to very similar plots for the tree-based models, so only the compressed kernel representation algorithm has been shown for conciseness. Note in the case of the ExtraTrees model, each tree was individually perturbed before being combined. Further, in the case of the perceptron, a GMM was fit to the data in the inferred latent space rather than in the original input space.

A downside to group fairness algorithms more generally, as pointed out in [105], is that candidate systems which impose group fairness can lead to qualified candidates being discriminated against. This can be visually verified as the perturbation pushes down the outcome of many orange points below the total population mean in order to satisfy the constraint. By choosing to incorporate group fairness constraints the practitioner should be aware of these tradeoffs.

4.8.2 ProPublica Dataset - Racial Biases

Across the USA, judges, probation and parole officers are increasingly using algorithms to aid in their decision making. The ProPublica dataset [1] contains data about criminal defendants from Florida in the United States. It is the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) algorithm

[1] https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis

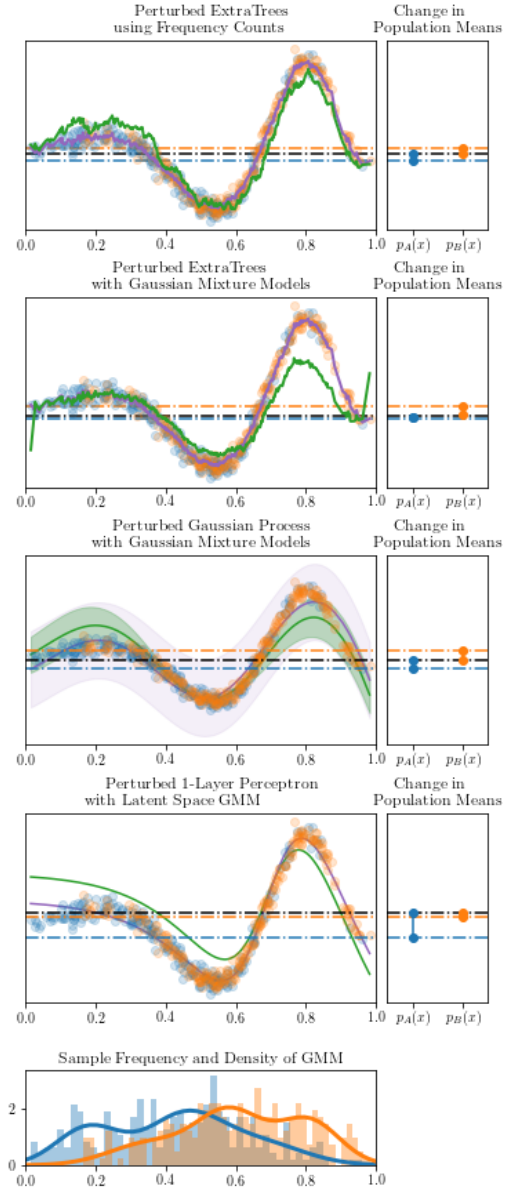


Figure 4.2: Above shows synthetic data of two populations, $p_A(x)$ (blue) and $p_B(x)$ (orange). The main plots show the observations and the perturbation to the respective models. Purple functions identify the original inferred functions and green indicates the fair perturbed inferred functions. Below the main plots show a normalised histogram of the observations for the $p_A(x)$ and $p_B(x)$ populations respectively along with the PDF of the Gaussian mixture model of their respective densities. To the right shows how the expected mean of the two populations have been perturbed to be equal.

[45] which is often used by judges to estimate the probability that a defendant will be a recidivist, a term used to describe re-offenders. However, the algorithm is said to be racially biased against African-Americans [47]. In order to highlight the proposed algorithm, we first endeavour to use a random forest to approximate the decile scoring of the COMPAS algorithm and then perturb each tree to remove any racial bias from the system.

The two subpopulations we consider constraining are thus African-American and non-African-American. We encode the COMPAS algorithms decile score into an integer between zero and ten such that minimizing L_2 perturbation is an appropriate objective function. The fact the decile scores are bounded in $[0, 10]$ was not taken into account. The random forest used 20 decision trees as base estimators and the explicit kernel representation version of the algorithm was used for the sake of demonstrative purposes.

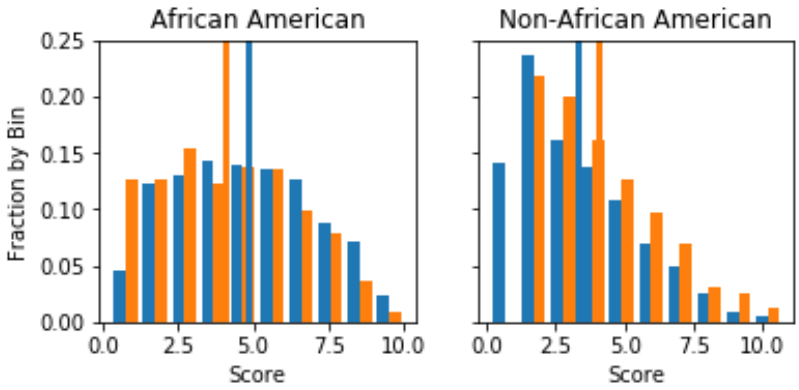


Figure 4.3: The above figure shows the output distribution of decile scores for African Americans and non-African Americans before (blue) and after (orange) the mean equality constraint was applied. We can see that the respective means (vertical lines) become approximately equal after the inclusion of the constraint using the empirical input distribution.

Figure 3 presents the marginal distribution of predictions on a 20% held out

test set before and after the GFE constraint was applied. It is visible that both the expected outcome for African Americans is decreased and for non-African Americans is increased. Notice that while the means are equal to the structure of the two of distributions are quite different, indicating that GFE constraints still allow greater flexibility than more strict group fairness such as that described in Section 1. The root square difference between the predicted points before and after perturbation was 0.8. Importantly, the GFE constraint described in this work was verified numerically with the average outputs recorded as,

	Unconstrained	Constrained
African Am.	4.82	4.41
non-African Am.	3.26	4.41

Table 4.3: Mean score before and after GFE perturbation.

4.9 Intersectionality

An important issue raised in [51], a model that satisfies conditional parity with respect to race and gender independently may fail to satisfy conditional parity with respect to the conjunction of race and gender. In the social science literature concerns about, potentially discriminated against, sub-demographics are referred to as intersectionality [108]. This section investigates extending the proposed fair decision tree model to allow for multiple constraints.

For the sake of conciseness this section will present results only for compressed representation. Thus, we will endeavor to minimize the perturbations induced on a per leaf basis, irrespective of the number of data points per leaf. The core difference between single and multiple constraints is that we can no longer use the arrowhead matrix lemma, instead, we must work out the update using the

block matrix inversion lemma. Importantly, $p_A(x)$ and $p_B(x)$ for each constraint are defined as the empirical distributions of each subgroup considered. This is an important point as small subgroups may have empirical distributions which are not good approximations of the true generative distributions and hence our constrained space for inference may not constrain predictions to equate accordingly.

Using the block matrix inversion lemma we find,

$$K^{-1} = \begin{bmatrix} (1+\sigma_n^2)\mathrm{I} & z \\ z^T & \bar{z}^T\bar{z} \end{bmatrix}^{-1} = \begin{bmatrix} (1+\sigma_n^2)^{-1}\mathrm{I} - (1+\sigma_n^2)^{-2} z \left(z^T\bar{z} - (1+\sigma_n^2)^{-1} z^T z\right)^{-1} z^T & -(1+\sigma_n^2)^{-1} z \left(z^T\bar{z} - (1+\sigma_n^2)^{-1} z^T z\right)^{-1} \\ -(1+\sigma_n^2)^{-1} \left(\bar{z}^T\bar{z} - (1+\sigma_n^2)^{-1} z^T z\right)^{-1} z^T & \left(z^T\bar{z} - (1+\sigma_n^2)^{-1} z^T z\right)^{-1} \end{bmatrix}$$

By simply inserting this into the kernel regression equation and noting that the elements of $\bar{z}$ are necessarily zero, the following update to the expected mean can be found as,

$$f(\bar{\mathrm{x}}) = \frac{1}{1+\sigma_n^2}\left(\mathrm{y}_j - z_j\left(z^T z\right)^{-1} z^T \mathrm{y}\right),$$

with z_j indicating the row of z relating to the difference of subgroup distributions on leaf j. The effect of the noise can be removed by post multiplying by $1+\sigma_n^2$.

4.10 Experiments: Intersectionality

4.10.1 ProPublica & the COMPAS System

The first experiment reproduces the experiment in Section 4.8.2 which uses a random forest to estimate the recidivism decile scores of the COMPAS algorithm applied to the ProPublica dataset while adding a GFE constraint between African Americans and Non-African Americans. However, it can also be noted that Hispanics also receive a similar discrimination. Figure 4.4, visualizes the effect of GFE

constraints on the predicted distributions of the three demographics.

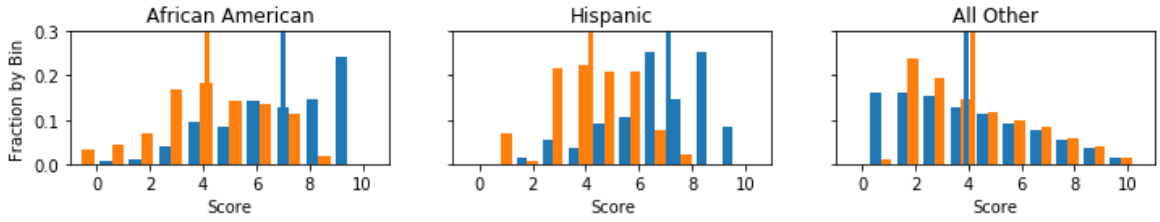


Figure 4.4: The figure shows the effect of GFE constraints in the inferred scores of the ProPublica dataset between African American, Hispanic and all other defendants. Before perturbation is in blue and after in orange. Vertical lines indicate the mean of the distributions.

4.10.2 Illinois State Employee Salaries

The Illinois state employee salaries[2] since 2011 can be seen to have a gender bias and bias between veterans and non-veterans. The motivation of this experiment was if one wished to predict a fair salary for future employees based on current staff. Gender labels were inferred using the employees' first names, parsed through the *gender-geusser* python library. GFE constraints were applied between all intersections of gender and veteran / non-veterans, the marginals of gender and the marginals of veteran / non-veterans. Table 1 shows the expected outcome of each group before and after GFE constraints are applied and Figure 3 visualizes the perturbations to the marginals of each demographic intersection due to the GFE constraints. The train-test split was set as 80%-20% and the incorporation of the GFE constraints increase the root mean squared error from $12,086 to $12,772, the cost of fairness, that is to say the loss in predictive performance in order to uphold the constraint.

[2] `https://data.illinois.gov/datastore/dump/1a0cd05c-7d17-4e3d-938d-c2bfa2a4a0b1`

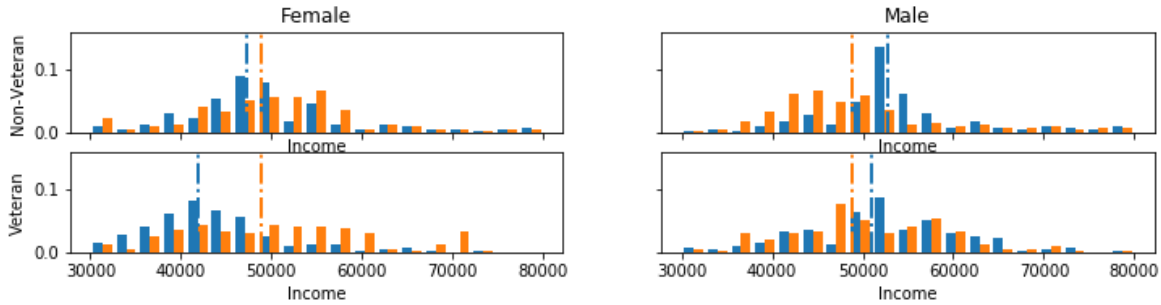


Figure 4.5: The figure visualizes the distribution of salaries before and after perturbations due to GFE constraints. It is clear that female veterans benefit the most from such a constraint, while male non-veterans lose out. Colors and lines denoting the same meaning as the previous figures. The figures are cropped to the main mode of salaries to facilitate visual comparisons.

Group	Female Non-Vet.	Male Non-Vet.	Female Vet.	Male Vet.	Male	Female	Vet.	Non-Vet.
Original	47,334	52,777	41,890	51,063	46,962	52,215	49,555	49,805
Perturbed	*48,695*	*48,693*	*48,694*	*48,693*	*48,695*	*48,698*	*48,775*	*48,775*

Table 4.4: The above table shows the expected outcome of a random tree regressor with and without GFE constraints applied to four sub-demographics, between gender and between veterans and non-veterans.

4.11 Conclusion

Regulatory bodies have shown precedent in developing affirmative action and other group fairness policy. This work develops group fairness constrained machine learning techniques. While relatively simple to understand and easy to incorporate into models used by practitioners, the methodology of this chapter has a direct impact to four of the ten top data science algorithms according to [166].

I believe this chapter is rich with novel contributions. Firstly, an open problem outlined in [46] has been addressed, that is how can group fairness be applied to regression problems. This work has outlined how this can be achieved for all kernel regression techniques which minimise the L2 loss of a regressor. Secondly, the chapter outlines an approach that allows practitioners to constrain inference such that integrals of the regressor with respect to defined densities are equal. I believe

this can have much greater application although have not yet identified other *killer applications* yet. Finally, this chapter draws upon the commonality between the various kernel methods, outlined in Chapter 3, in order to create a very general approach. I have shown that this does not hinder the computational or memory efficiency of popular and fast inference techniques such as decision trees. All three of these aspects adds to the works novelty and a variant of this work has been published in my paper at Entropy, 2019 as discussed in Chapter 1.

Part III

Overcoming Scalability Issues

Chapter 5

Scaling Gaussian Processes

Gaussian processes and other kernel methods have a wealth of applications, but their computational and memory complexity is such that they can often not be applied naïvely in practice. As such, the community has gone to great lengths to derive scalable approaches to Gaussian process, the richness of which could lead to an entire book. This chapter endeavours to highlight *some* of the major ideas and approaches that have been developed over the past two decades and hopefully will serve as a context for the coming chapters.

5.1 Parametric Sparse Gaussian Processes

The majority of sparse Gaussian Process techniques involve parametric approximations of the otherwise non-parametric regression model. These approximate Gaussian Processes are sometimes referred to as semiparametric machine learning techniques. In this section, an overview of some of the prominent methods are presented and an analysis of the effect of the approximation is discussed. Endeavouring to remain concise, this section will focus on only the predominant techniques as seen in the review papers [164, 126, 37].

In order to speed up Gaussian Processes to a reasonable computational com-

plexity, ideally to linear time complexity, the inversion of the covariance matrix, $(K + \sigma^2_{noise} I)^{-1}$, is often approximated in some form. One approach to do this is to approximate the covariance matrix with a matrix of reduced rank.

Let K be a matrix of rank q then $K = U\Sigma V^T$, where Σ is a diagonal matrix in $R^{q \times q}$ and U, V are unitary rotation matrices. These are the eigenvalues and eigenvectors of K. As such the inversion of K can be computed by finding $V\Sigma^{-1}U^T$. As inversion is restricted only to Σ the computational complexity is reduced to $O(nq^2)$.

When K is not of rank q, the optimal low rank approximation to K can be found by truncating the eigenspectrum to the first q eigenvalues and eigenfunctions,

$$\bar{K} = \underset{rank(\hat{K}) \leq q}{argmin} \; ||K - \hat{K}||, \tag{5.1}$$

when,

$$\bar{K} = U_q \Sigma_q V_q^T. \tag{5.2}$$

Unfortunately, performing singular value decomposition, which is required to decompose K into this form, costs $O(n^3)$ and hence is too computationally expensive to be used in practice. For finding only a very small number of eigenvalues and eigenfunctions the power iteration can be used iteratively which costs $O(n^2)$ per iteration. However, this is also too computationally expensive to be used for most practical use cases.

Under the reproducing kernel Hilbert space (RKHS) view of the GP, we can see K as being constructed by the product of two data matrices, $K = X^T X$. This exactly follows from the definition of a kernel as the dot product between data points in some feature space, $\phi(x)$. As such X is in $R^{q_\phi \times n}$ where q_ϕ is the dimensionality of

the feature space and n is the number of data points. Note that q_ϕ may be infinity. Using this new definition we can decompose X into its eigenspectrum,

$$X = U_x \Sigma_x V_x^T. \tag{5.3}$$

As such

$$K = X^T X = (V_x \Sigma_x U_x^T)(U_x \Sigma_x V_x^T) = V_x \Delta V_x^T, \tag{5.4}$$

where $\Delta = \Sigma_x^2$. By letting U_q be the first q columns of U_x then the optimal approximation is equivalently given by

$$K = X^T P X, \tag{5.5}$$

where $P = U_q U_q^T$. This can be interpreted as projecting the data onto a low dimensional subspace in R^q.

These two views of the optimal low-rank approximations of the covariance matrix have motivated a number of sparse Gaussian Process methods. This section is entitled parametric sparse approximations as Gaussian Processes are traditionally thought of as being non-parametric, or infinitely parametric, but as we limit the eigenspectrum of K and as such the degrees of freedom of the Gaussian Process we retrieve a parametric or semi-parametric technique. Since the celebrated result of Johnson and Lindenstrauss [89], which showed that random low-dimensional embeddings preserve Euclidean geometry, there has been a great deal of work on low-rank linear algebra. Much of this has spread to the machine learning community and as such sparse parametric Gaussian Processes tend to be the most common of the scalable Gaussian Processes in the literature.

5.1.1 Subset of the Data Points (SoD)

The Subset of Data is the simplest and perhaps most trivial method to reduce the complexity of Gaussian Processes. If the computational complexity of the Gaussian Processes is restricted to $O(m^3)$ then the method simply selects m datapoint from the set of n datapoint to train. Therefore we deal only with a rank m matrix approximating the otherwise rank n matrix. To find the optimal m data points would require testing n choose m data points and thus would be too computationally expensive. As such a random subset is usually taken. As pointed out in [126], the computational complexity of SoD is constant, rather than linear in n, as such there may be interesting ways to select the m data points using some active selection scheme. However, in this review, we will not consider advancing methods but rather reviewing the current literature.

5.1.2 Nyström's Method

Similarly, if K is full rank, Nyström's method [163, 60] uses a subset of q columns and rows of K with which the eigenvalues and vectors can be found. These eigenvalues and eigenvectors are used to create a low-rank approximate covariance matrix $\tilde{K}$.

More formally let

$$K = \begin{bmatrix} K_{1,1} & K_{1,2}^T \\ K_{1,2} & K_{2,2} \end{bmatrix}, \tag{5.6}$$

be a partitioning of the covariance matrix K. Further let $\bar{K}_{1,1}$ be the optimal rank q approximation of $K_{1,1}$. Then the Nyström approximation to the inverse of K is

$$K_{\text{Nyström}} = [K_{1,1}, K_{1,2}]^T \bar{K}_{1,1}^{-1} [K_{1,1}, K_{1,2}]. \tag{5.7}$$

Another view of Nyström's method is to look at the eigenspectrum. The eigenvalues and vectors of the approximate submatrix are extended to a matrix of size $n \times n$ by

$$\Sigma_{\text{Nyström}} = \frac{n}{q}\bar{\Sigma}_{1,1}, \tag{5.8}$$

$$U_{\text{Nyström}} = \sqrt{\frac{q}{n}} C \bar{U}_{K_{1,1}} \bar{\Sigma}_{1,1}^{-1}. \tag{5.9}$$

Nyström's method can also be seen as creating an exact Gaussian Process with an approximate kernel which leads to the low rank, or degenerate, covariance matrix $\tilde{K}$. That is the kernel becomes

$$\kappa_{\text{Nyström}}(x_i, x_j) = \kappa(x_i, x_u)^T \kappa(x_u, x_u)^{-1} \kappa(x_u, x_j), \tag{5.10}$$

where x_u relate to a fixed number of bases defined by $\bar{K}_{1,1}$.

The selection of the subdivision of the matrix is not trivial and uniform random sampling has been used commonly. Other distributions used to select submatrices have been presented, such as diagonal sampling and column norm sampling. The intuition behind diagonal sampling is to select submatrices which would otherwise have high variance. As such we sample columns based on the unnormalised discrete distribution defined by the diagonal values $K_{i,i}$. The column norm sampling regime samples a submatrix based on the L_2 norm of the columns. Again this can be seen intuitively as selecting a submatrix that approximately minimises the variance of the remaining partition of K. Both regimes use heuristics to attempt to improve on random sampling but have additional computational costs, $O(n)$ for diagonal sampling and $O(n^2)$ for column norm sampling. There are also a great number of active sampling regimes such as Incomplete Cholesky Factori-

sation (ICL) and Adaptive Nyström Sampling. For further details on Nyström's method and sampling regimes the reader is directed to [100, 3].

5.1.3 Subset of Regressors (SoR)

The Subset of Regressors approach [146, 126] endeavours to force the information between the training and testing points through a set of m auxiliary points u,

$$p(f_*|y) \approx \mathcal{N}(Q_{*,f}(Q_{f,f} + \sigma^2 I)^{-1} y, Q_{*,*} - Q_{*,f} Q_{f,f}^{-1} Q_{f,*}), \tag{5.11}$$

where $Q_{a,b} = K_{a,u} K_{u,u}^{-1} K_{u,b}$. This can be seen as an exact Gaussian Process with an approximate kernel,

$$\kappa(x_a, x_b) \approx \kappa(x_a, x_u) K_{u,u}^{-1} \kappa(x_u, x_b). \tag{5.12}$$

As we force the relationship between the training and testing via a set of m inducing points, we limit the model to m degrees of freedom. This is to say that there are only m linearly independent functions which can be drawn from the prior. Further, as we may get a large number of training points with little observation noise, the matrix inversion will become increasing poorly conditioned.

5.1.4 Deterministic Training Conditional (DTC)

Originally presented by [142] under the name Projected Latent Variable and later referred to as Projected Process Regression [164], the DTC method approximates the likelihood using a fixed number of bases,

$$p(y|f) \approx q(y, u) = \mathcal{N}(K_{f,u} K_{u,u}^{-1} u, \sigma^2 \mathbf{I}), \tag{5.13}$$

where u are a set of auxiliary or pseudo data points. The DTC however does use the full exact covariance matrix on the test points. That is to say the covariance matrix is partitioned between the columns which correspond to the training points and auxiliary points,

$$\tilde{K} = \begin{bmatrix} Q_{f,f} & Q_{f,*}^T \\ Q_{f,*} & K_{*,*} \end{bmatrix}, \tag{5.14}$$

where $Q_{a,b} = K_{a,u}K_{u,u}^{-1}K_{u,b}$. By making this assumption the predictive distribution becomes,

$$q_{DTC}(f_*|f) = \mathcal{N}(\sigma^{-2}K_{*,u}\Sigma K_{u,f}^{-1}y, K_{*,*} - Q_{*,*} + K_{*,u}^T\Sigma K_{*,u}), \tag{5.15}$$

where $\Sigma = (\sigma^{-2}K_{u,f}K_{f,u} + K_{u,u})^{-1}$. Intuitively this can be seen as squashing the training points through a set of auxiliary points u. As such the rank of $Q_{u,u}$ is limited to q, where q is the number of data points in u. However, as opposed to the SoD and Nyström's method these pseudo points do not have to be observed, rather, they can be superpositions of observed points. This gives pseudo point models an additional level of freedom which is accounted for in the additional computational cost of optimising over the pseudo points positions in the input space. Note though that the pseudo points u do not directly rely on the observed outputs y for kernels that are independent of the y values, which is the great majority of cases. However, through the optimisation of the pseudo points we usually minimise the negative log marginal likelihood of the Gaussian Process, which does take into account the data fit, that is the probability the data was generated from the Gaussian Process given the set of observations. This clearly distinguishes the approach from SoD and Nyström's method which are based solely on approximating the covariance matrix irrespective of the y values

As the covariance of the training and test data points are treated differently, DTC cannot be seen as an exact Gaussian Process.

A variant to DTC is the Augmented DTC, or ADTC, which augments the testing points to the set of auxiliary data points. This can also be done to the Subset of Regressors method (see Section 3.1) creating ASoR. These two methods are equivalent.

5.1.5 Fully Independent Training Conditional (FITC) and Partially Independent Training Conditional (PITC)

FITC, first introduced as Sparse Gaussian Processes using Pseudo-inputs [147], builds on the DTC approach with a richer likelihood approximation

$$p(y|f) \approx q(y,u) = \mathcal{N}(K_{f,u}K_{u,u}^{-1}u, diag[K_{*,*} - Q_{*,*}] + \sigma^2\mathbf{I}), \tag{5.16}$$

where *diag* indicates the diagonal matrix which takes on the diagonal values of $K_{*,*} - Q_{*,*}$. As such, the effective prior can be denoted as

$$\tilde{K} = \begin{bmatrix} Q_{f,f} - diag[Q_{f,f} - K_{f,f}] & Q_{f,*}^T \\ Q_{f,*} & K_{*,*} \end{bmatrix}. \tag{5.17}$$

The sole difference between DTC and FITC is the above rectification of the marginal likelihoods or diagonal values of the covariance. Despite this benefit, FITC has the same computational complexity as DTC. By simply propagating this through to the posterior distribution over f_* we arrive at

$$q_{DTC}(f_*|f) = \mathcal{N}(K_{*,u}\Sigma K_{u,f}\Delta^{-1}y, K_{*,*} - Q_{*,*} + K_{*,u}^T\Sigma K_{*,u}), \tag{5.18}$$

where $\Delta = diag[K_{f,f} - Q_{f,f}]$ and $\Sigma = (K_{u,f}\Delta^{-1}K_{f,u} + K_{u,u})^{-1}$.

PITC is another extension of FITC which simply exchanges the diagonal rectification of $diag[Q_{f,f} - K_{f,f}]$ with a block diagonal variant $block_diag[Q_{f,f} - K_{f,f}]$. This allows us to have sets of points which are independent again. Further, both FITC and PITC can be seen as full Gaussian Processes with approximate kernels, if we treat the test points the same as our training points as opposed to partitioning our covariance matrix.

5.1.6 Random Fourier Features

Random Fourier feature approaches [129] have become a widely popular approach to the scaling of many station kernel inference, including but not exclusive to Gaussian processes. The concept is easily digestible for users, yet highly effective. The premise is to approximate non-degenerate kernels with a degenerate alternative by explicitly defining a Hilbert space for the model. The choice of feature map is chosen in order to maintain stationarity, hence Fourier components are chosen. Kernel priors $K(x, x^*)$ are transformed into a prior over the Fourier space and frequencies, θ, are sampled using a Monte Carlo approach. For each θ_i selected two explicit features are created, namely $\phi_i(x) = \cos(\theta x)$ and $\phi_{i+1}(x) = \sin(\theta x)$. When the inner product of these is computed we retrieve a stationary approximate kernel of the form $K(x, x^*) \approx \frac{1}{D}\sum_{i=1}^{D} \cos(\theta_i(x - x^*))$. As these Fourier components were sampled from the prior distribution they do not need to be weighted.

5.2 Local Expert Models

Local expert models endeavour to combine the predictions of many Gaussian Processes, each of which only utilises a subset of the data. The most common is a multiplicative model, that is the Product of Experts, generalised Product of Experts,

Bayesian Committee Machines and robust Bayesian Committee Machines.

5.2.1 Multiplicative methods

Two popular schemes to do multiplicative model combination are the Product of Experts (PoEs) model [114, 35] and the Bayesian Committee Machines (BCMs) [151, 43]. The latter can be seen as an advancement on the simpler PoE model. While there has been a number of incremental advances on these two core methodologies in recent years the principals which will be discussed in this chapter are general enough to be extended to these. Unlike the previous methods discussed, multiplicative model combination are transductive meaning that how models are combined depends on the points to be inferred as opposed to solely on the training data.

Product of Experts (PoEs) The Bayesian Committee Machine is a transductive approximation to a Gaussian Process. The goal is to divide the data into m sets and train a GP on each of these sets independently. The predictions of the GPs are combined as follows

$$\mu_{PoE} = \sigma_*^2 \sum_{i=1}^{n} \sigma_i^{-2} \mu_i, \tag{5.19}$$

$$\sigma_{PoE}^{-2} = \sum_{i=1}^{n} \sigma_i^{-2}. \tag{5.20}$$

The Subscript i indexes the mean and variance of the i^{th} local expert and σ_{**}^2 is the prior variance. The above equations simply relate to multiplying the GP predictions together

$$P(f^*|x^*, D)_{PoE} = \prod_{i=0}^{k} P(f^*|x^*, D^{(i)}). \tag{5.21}$$

Bayesian Committee Machines (BCMs) The Bayesian Committee Machine is another transductive approximation which builds on PoEs while removing the effect of combining the prior multiple times. The GPs are referred to as local experts and are recombined using a Bayesian optimal classifier as

$$\mu_{BCM} = \sigma_*^2 \sum_{i=1}^{n} \sigma_i^{-2} \mu_i, \tag{5.22}$$

$$\sigma_{BCM}^{-2} = \sum_{i=1}^{n} \sigma_i^{-2} - (m-1)\sigma_{**}^{-2}, \tag{5.23}$$

where $\mu_{BCM}, \sigma_{BCM}^2$ are the mean and variance of the prediction. The BCM has a similar formulation to the PoE, but with a normalisation term included to take into account the accumulation of the prior

$$P(f^*|x^*, D)_{BCM} = \frac{\prod_{i=0}^{m} P(f^*|x^*, D^{(i)})}{P(f^*|x^*)^{1-m}}. \tag{5.24}$$

For consistency through this chapter, we will now reformulate the PoE and BCM to a standard notation which will be useful later. Let σ_i^2 and $\mu_i(x)$ be the variance and mean of the prediction of the i^{th} local expert. Let us define Σ as the joint covariance matrix between all the independent local experts

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m^2 \end{pmatrix}. \tag{5.25}$$

Further, let C be the associated correlation matrix of Σ. Using this notation and letting μ be the vector of $\mu_i(x)$ mean predictions, we define the predictions of the PoE model as

$$\mu_{PoE} = (1^T \Sigma^{-1} 1)^{-1} 1^T \Sigma^{-1} \mu_i, \tag{5.26}$$

$$\sigma_{PoE}^{-2} = (1^T \Sigma^{-1} 1)^{-1}. \tag{5.27}$$

In a similar fashion, we can define the BCM predictions as

$$\mu_{BCM} = (1^T\Sigma^{-1}1 - (1^TC^{-1}1 - 1)\sigma_{**}^{-2})^{-1}1^T\Sigma^{-1}\mu_i, \tag{5.28}$$

$$\sigma_{BCM}^{-2} = (1^T\Sigma^{-1}1 - (1^TC^{-1}1 - 1)\sigma_{**}^{-2})^{-1}. \tag{5.29}$$

More recently variants of these techniques have emerged which use heuristics in order to weight the predictions of local experts in a greedy fashion. The Generalised Product of Experts (gPoE) [35] and Robust Bayesian Committee Machine (rBCM) [43] vary from the PoE and BCM models by reweighing the covariance matrix between experts as

$$\Sigma = \begin{pmatrix} \sigma_1^2\beta_1^{-1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_m^2\beta_m^{-1} \end{pmatrix} \tag{5.30}$$

where β are the weights. This corresponds to weighing the experts as follows

$$P(f^*|x^*, D)_{PoE} = \prod_{i=0}^{k} P^{\beta_i}(f^*|x^*, D^{(i)}), \tag{5.31}$$

$$P(f^*|x^*, D)_{BCM} = \frac{\prod_{i=0}^{m} P^{\beta_i}(f^*|x^*, D^{(i)})}{P(f^*|x^*)^{-1+\sum_i \beta_i}}. \tag{5.32}$$

As such C becomes a diagonal matrix of β_i^{-1} for all $i \in [1, m]$. The weights, β, are defined as the differential entropy between the prior and the predictive variance of the local expert,

$$\beta_i = \frac{1}{2}(\log\sigma_{**}^2 - \log\sigma_i^2(x_*)). \tag{5.33}$$

The motivation for these variants is to remove 'wiggles' and unwanted artefacts in the posterior distribution that are introduced by the respective multiplicative model combination schemes. It has been found that this reweighing of the local

experts suppresses their ability to extrapolate confidently which is the scenario where the unwanted artefacts tend to occur. However, it is worth noting that this reweighing is based on a heuristic and offers no clear probabilistic motivation but has good imperial performance. Further, the reweighing of the local expert predictions effects their ability to extrapolate. This can be shown in a simple example using two local experts each having one data point as seen in Figure 2.

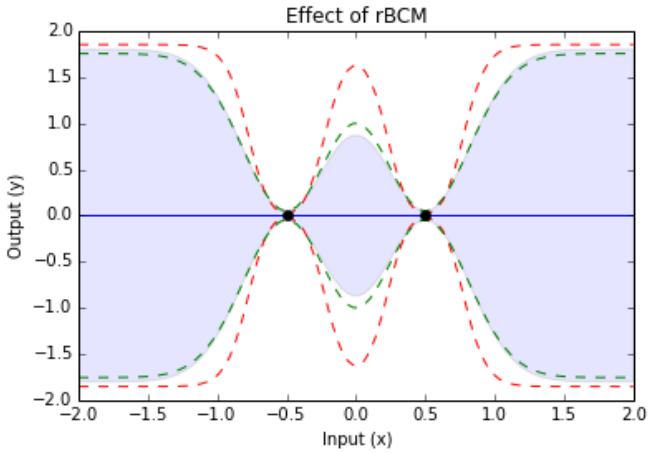


Figure 5.1: A comparison of the variance of the BCM (green), rBCM (red) and true Gaussian Process which is being approximated (blue). Each local expert has a single data point.

5.2.2 Treed Gaussian Processes

Treed Gaussian Processes use a combination of space partitioning trees and Gaussian Processes [72, 71]. The tree is a recursive partition over the input space, X, creating R non-overlapping regions, $\{r_v\}_{v=1}^{R}$ such that $X = \bigcup_{v=1}^{R} r_v$. Each of these r_v partitions contains a subset of data $\mathcal{D}_v = \{X_v, y_v\}$ with n_v observations. Using the $\mathcal{D}_v$ for each r_v, a Gaussian Process is constructed. Predictions are made based on the partition where the test point belongs.

Treed Gaussian Process have a clear reliance on the tree structure and the point of the boundary of the leaves are discontinuous. As such it is suggested [72] that the Treed Gaussian Process averages over tree partitions using a Monte Carlo approach.

A natural advantage of Treed Gaussian Processes is their ability to learn non-stationary functions as each leaf of the tree can learn either the kernel or kernel hyperparameters. There is also a clear link between the Treed Gaussian Process and the multiplicative local expert models after clustering has been applied.

5.3 Other Methods

5.3.1 Toeplitz Covariance Matrices

A particular type of matrix, namely a Toeplitz matrix, has structure such that $A_{i,j} = A_{i+1,j+1} = a_{i-j}$. The general Toeplitz matrix is invertible $O(n^2)$ and often does not need sparse approximation applied.

$$K = \begin{bmatrix} a_0 & a_1 & \dots & a_n \\ a_{-1} & a_0 & \dots & a_{n-1} \\ \vdots & \ddots & \ddots & \vdots \\ a_{-n} & a_{-n+1} & \dots & a_0 \end{bmatrix}. \tag{5.34}$$

This form of covariance matrix naturally appears in univariate time series where data comes in on regular fixed intervals. However, even if the data suffers from missing data or multiple frequencies of observations we can portion the matrix based on these constraints and use the block matrix inversions lemma. While this is not a Gaussian Process approximation, it has been included in this literature review due to its relevance in the fast computation of Gaussian Processes.

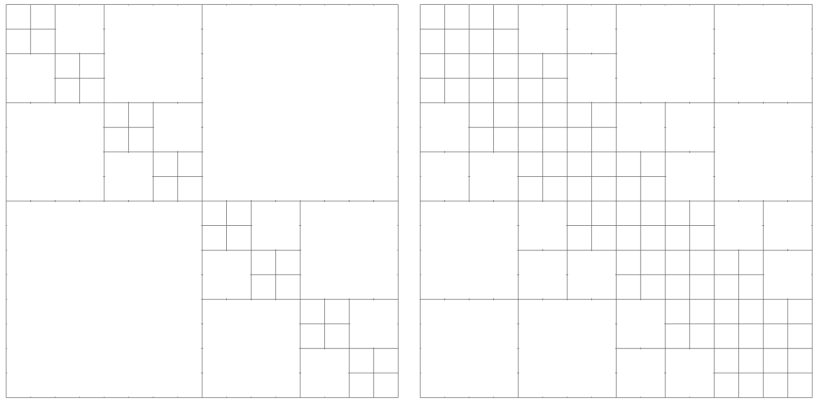


Figure 5.2: Some matrix partitions which can be represented as hierarchical matrices.

5.3.2 Hierarchical Matrices and Fast Multipole Methods

Another approach to sparse Gaussian Processes revolves around the usage of a hierarchical matrix, in particular, $\mathcal{H}^2$-matrix, approximation [21] of the covariance matrix of our Gaussian Process. $\mathcal{H}^2$-matrices are a data-sparse matrix approximation commonly used to deal with demands in integral equations and elliptical partial differential equations [21]. They build on two core ideas. The first is that a matrix may be segmented into many smaller submatrices. The second is that these sub-matrices may be efficiently stored using low-rank approximations.

More specifically, $\mathcal{H}^2$-matrices are based on hierarchically structured block partitions and low-rank representation of submatrices. In practice, this is found via taking the first m eigenvalues and eigenvectors per block which corresponds to the optimal low-rank approximation per block [27]. The construction of the block partition for $\mathcal{I} \times \mathcal{J}$ requires hierarchical partitions of the index sets $\mathcal{I}$ and $\mathcal{J}$. There is a naturally occurring trade-off between the depth of the hierarchy, the allowed error ϵ and computational and storage costs. This leads to a larger level of control

over the approximation for the user. Special cases of $\mathcal{H}^2$-matrices offer log-linear computational cost for matrix inversion and data storage.

Hierarchical matrices appear to be highly underrepresented in the Gaussian Process literature, we could only find one paper [20] which identified their use for Gaussian Processes and a small number of others which looked at their usage to approximate the more general covariance [103, 139, 76]. However, this leaves ample opportunity for further future research.

Fast Multipole Methods deal with a special case of $\mathcal{H}^2$-matrices, which are claimed to be created by stationary kernels such as exponential, Gaussian, rational quadratic and Matern kernels [1]. Recently there has been a number of numerical analysis papers which show linear time fast direct solvers [1, 38]. While these approaches have not yet become mainstream, there has already been public code made available.

5.4 Conclusion

This chapter presents an overview wide variety of approaches aimed at scaling Gaussian processes to large datasets. As can be seen the sheer volume of work in this domain is large, with each direction of research offering a set of pros and cons. In the coming chapters, I will outline a novel approach to scalable linear algebraic operations with a aim to develop a novel direction for future research.

Chapter 6

Implicit Matrix Approximation

A common theme throughout this thesis is the continued strife caused by linear algebra. While linear algebra offers a concise way to represent many systems of equations, the common operations performed on matrices are expensive in terms of computationally and memory. Some such operations include,

Matrix Inversion	$\mathcal{O}(n^3)$
Matrix Determinant	$\mathcal{O}(n^3)$
Matrix Powers	$\mathcal{O}(n^3)$
Matrix-Matrix Multiplication	$\mathcal{O}(n^3)$
Matrix-Vector Multiplication	$\mathcal{O}(n^2)$
Vector-Vector Multiplication	$\mathcal{O}(n)$

However, as seen in the previous chapter there are a number of approximations that are used in practice to allow kernel methods to be used which both reduce the amount of memory required to store the matrix and reduce the computational complexity of matrix operations.

Generally speaking, the approaches focussed on accelerating kernel methods use 'sparse' matrix techniques. For those outside the kernel community, this sometimes seems anti-intuitive as the kernel matrix remains dense. However, this term refers to low-rank approximations which are sparse in the number of basis functions they use.

A natural question to then ask is "Why don't they just use sparse matrices like

the rest of us?". This is a valid question and I believe has a reasonable set of pro's and con's to stir up a good debate. On the one hand, s-sparse matrices only require $\mathcal{O}(ns)$ memory and matrix-vector multiplication requires only $\mathcal{O}(ns)$ computational operations.

Some sparse matrices may occur naturally in kernel methods. Gauss Markov random fields, or GMRF's, are Markov fields which interact via Gaussian interactions. As such the equations which model their relationship are identical to a Gaussian process with the distinction that their precision matrix is naturally sparse. Usually, the model is modelled by the precision matrix [102].

In many cases, though the kernel matrix is naturally dense. The question then becomes whether or not to force sparsity to accelerate computation. This is somewhat a double-edged sword. From one point of view, we can bound the error induced on the results by using results from matrix perturbation theory. On the other hand, we can view this from the kernel theory point of view. If one was to suppress all small elements of a kernel matrix this would be equivalent to approximating the kernel function to be a function with finite support. Unfortunately, this leads to issues in high dimension spaces as perturbations of the range of the kernel can lead to drastically different results. Some naturally compact kernels, such as circular and spherical kernels, and their advantages are outlined by Genton [63].

In this chapter, we detail an technique to approximate the trace of high dimensional sparse matrices, namely stochastic trace estimation. Using this method we show how we can estimate the trace of powers of sparse matrices which allows us to estimate the log determinant of kernel matrices and perform counting operations on adjacency matrices. This technique is used throughout the coming chapters.

6.1 Stochastic trace estimation

The problem of stochastic trace estimation is relevant to a range of problems from physics and applied mathematics such as electronic structure calculations [11], seismic waveform inversion [156], discretized parameter estimation problems with PDEs as constraints [75] and approximating the log determinant of symmetric positive semi-definite matrices [23]. Machine learning, in particular, is a research domain which has many uses for stochastic trace estimation. They have been used efficiently by Generalised Cross Validation (GCV) in discretized iterative methods for fitting Laplacian smoothing splines to very large datasets [86], computing the number of triangles in a graph [4, 5], string pattern matching [8, 152] and the training Gaussian Processes using score functions [150]. Motivated by accelerating Gaussian graphical models, Markov random fields, variational methods and Bregman divergences work based on stochastic trace estimation has also been developed to improve the computational efficiency log-determinant calculations [77].

Stochastic trace estimation endeavours to choose n-dimensional vectors $\mathbf{x}$ such that the expectation of $\mathbf{x}^T A\mathbf{x}$ is equal to the trace of the implicit symmetrical positive semi-definite matrix $A \in \mathbb{R}^{n\times n}$. It can be seen that many sampling policies satisfy this condition. Due to this, several metrics are used in order to choose a sampling policy such as the single shot sampling variance, the number of samples to achieve a (ϵ,δ)-approximation and the number of random bits required to create $\mathbf{x}$ [9]. This last metric is motivated in part by the relatively long timescales for hardware number generation, and concerns about parallelising pseudo-random number generators.

The novel aspect of this chapter is introduced as we propose a new stochastic trace estimator based on mutually unbiased bases (MUBs) [141], and quantify

the single shot sampling variance of the proposed MUBs sampling method and its corresponding required number of random bits. We will refer to methods which sample from a fixed set of basis functions as being fixed basis sampling methods. For example, we can randomly sample the diagonal values of the matrix A by sampling $\mathbf{x}$ from the set of columns which form the identity matrix. This is referred to as the unit vector estimator in the literature [9]. Other similar methods sample from the columns Discrete Fourier Transform (DFT), the Discrete Hartley Transform (DHT), the Discrete Cosine Transform (DCT) or a Hadamard matrix. We prove that sampling from the set of mutually unbiased bases significantly reduces this single shot sample variance, in particular in the worst case bound.

6.2 Mutually unbiased bases

Linear algebra has found application in a diverse range of fields, with each field drawing from a common set of tools. However, occasionally, techniques developed in one field do not become well known outside of that community, despite the potential for wider use. In this work, we will make extensive use of mutually unbiased bases, sets of bases that arise from physical considerations in the context of quantum mechanics [141] and which have been extensively exploited within the quantum information community [49]. In quantum mechanics, physical states are represented as vectors in a complex vector space, and the simplest form of measurement projects the state onto one of the vectors from some fixed orthonormal basis for the space, with the probability for a particular outcome given by the square of the length of the projection onto the corresponding basis vector [1]. In such a setting, it is natural to ask about the existence of pairs or sets of measurements

[1]For a more comprehensive introduction to the mathematics of quantum mechanics in finite-dimensional systems, we refer the reader to [116]

where the outcome of one measurement reveals nothing about the outcome of another measurement, and effectively erases any information about the outcome had the alternate measurement instead been performed. As each measurement corresponds to a particular basis, such a requirement implies that the absolute value of the overlap between pairs of vectors drawn from bases corresponding to different measurements be constant. This leads directly to the concept of mutually unbiased bases (MUBs).

A set of orthonormal bases $\{B_1, \ldots, B_n\}$ are said to be mutually unbiased if for all choices of i and j, such that $i \neq j$, and for every $\mathbf{u} \in B_i$ and every $\mathbf{v} \in B_j$, $|\mathbf{u}^\dagger \mathbf{v}| = \frac{1}{\sqrt{n}}$, where n is the dimension of the space. While for real vector spaces the number of mutually unbiased bases has a complicated relationship with the dimensionality [26], for complex vector spaces the number of mutually unbiased bases is known to be exactly $n + 1$ when n is either a prime or an integer power of a prime [98]. Furthermore, a number of constructions are known for finding such bases [98]. When n is neither prime nor a power of a prime, the number of mutually unbiased bases remains open, even for the case of $n = 6$ [32], but is known to be at least $p_1^{d_1} + 1$, where $n = \prod_i p_i^{d_i}$ and p_i are prime numbers such that $p_i < p_{i+1}$ for all i.

One practical method for constructing MUBs is to use the unitary operators method with finite fields [15], which is effective when the dimensionality of the space is either prime or a prime power. For conciseness, we will outline the procedure for only the prime dimensionality case but note that any integer dimensional space is at most bounded by two times its closest prime power dimension which adds a constant cost to the memory and runtime performance. First, let us construct the matrix X as the identity matrix with the columns shifted one to the left creating the form,

$$X = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \end{bmatrix}$$

and letting Z be a diagonal matrix with elements set to the roots of unity, $Z_{k,k} = \exp\left(\frac{2k\pi i}{n}\right)$. Given these two matrices a set of mutually unbiased bases are found as the eigenvectors of the matrices,

$$X, Z, XZ, XZ^2, \dots, XZ^{n-1}.$$

At first glance it may appear that the computational cost of constructing vectors from these bases is $\mathcal{O}(d^3)$ due to the cost decomposing these matrices in $\mathbb{C}^{n\times n}$, however, under more scrutiny we can see that X is a circulant permutation matrix and as such the elements its eigenvectors are equal to $U_{k,j} = \frac{1}{\sqrt{n}} \exp\left(\frac{jk2\pi i}{n}\right)$ irrespective of the dimensionality, where j indexes the elements of the eigenvector and v indexes which eigenvector is under consideration.

As the elements the diagonal matrix Z are the roots of unity in ascending order, it can be seen that $\exp\left(\frac{2\pi i}{n}\right)^k Z^k X = Q^k X = XZ^k$, where Q is some matrix of the same form as Z but with a shift of phase of the non-zero elements. As such, by writing the eigenbasis of $X = U^{-1}\Sigma U$ we can derive the eigenbasis of XZ^k for arbitrary value k with eigen decomposition $XZ^k = \hat{U}^{-1}\hat{\Sigma}\hat{U}$,

$$\begin{aligned} XZ^k &= U^\dagger \Sigma U Z^k \\ &= Q^k U^{-1} \Sigma U \end{aligned}$$

Next, we pull $Q^{\frac{k}{2}}$ and $Z^{\frac{k}{2}}$ through the eigenvectors by observing that we can transform the eigenvectors as $\Sigma = Z^{\frac{1}{2}}\hat{\Sigma}Q^{\frac{k}{2}}$,

$$\begin{aligned} XZ^k &= Q^{\frac{k}{2}}U^{-1}\Sigma U Z^{\frac{k}{2}} \\ &= Q^{\frac{k}{2}}U^{-1}Z^{\frac{1}{2}}\hat{\Sigma}Q^{\frac{k}{2}}UZ^{\frac{k}{2}} \\ &= \hat{U}^{-1}\hat{\Sigma}\hat{U} \end{aligned}$$

where $\hat{U} = Q^{-\frac{k}{2}}Z^{\frac{k}{2}}U$ and hence the $\hat{U}_{i,j} = \frac{1}{\sqrt{n}}\exp\left(\frac{2\pi i}{n}(jv + \frac{(j+1)(j+2)}{2}k)\right)$ using the same indexing as before.

As a result, we can simply use the following procedure to sample the vector $\mathbf{x}$ in linear computational time and memory:

1 Choose k and v, representing the basis and the vector to select respectively, uniformly at random.

2 If $k = 0$, then we select the vector v from the computational basis, that is to say the columns of the identity matrix.

3 Else, let $x_j = \frac{1}{\sqrt{n}}\exp\left(\frac{2\pi i}{n}(jv + \frac{(j+1)(j+2)}{2}k\right)$

6.3 A novel approach to trace estimation

In order to estimate the trace of a $n \times n$ positive semi-definite matrix A from a single call to an oracle for $\mathbf{x}^\dagger A\mathbf{x}$, we consider four strategies:

- **Fixed basis estimator:** For a fixed orthonormal basis B, choose $\mathbf{x}$ uniformly at random from the elements of B. The trace is then estimated to be $n\mathbf{x}^\dagger A\mathbf{x}$.

- **Mutually unbiased bases (MUBs) estimator:** For a fixed choice of a set of b mutually unbiased bases $\mathbb{B} = \{B_1, ..., B_b\}$, choose B uniformly at random from $\mathbb{B}$ and then choose $\mathbf{x}$ uniformly at random from the elements of B. Here b is taken to be the maximum number of mutually unbiased bases for a complex vector space of dimension n. As in the fixed basis strategy, the trace is then estimated to be $n\mathbf{x}^\dagger A\mathbf{x}$.

- **Hutchinson's estimator:** Randomly choose the elements of $\mathbf{x}$ independently and identically distributed from a Rademacher distribution $\left(Pr(x_i = \pm 1) = \frac{1}{2}\right)$. The trace is then estimated to be $\mathbf{x}^\dagger A\mathbf{x}$.

- **Gaussian estimator:** Randomly choose the elements of $\mathbf{x}$ independently and identically distributed from a zero mean unit variance Gaussian distribution. The trace is then estimated to be $\mathbf{x}^\dagger A\mathbf{x}$.

The first strategy is a generic formulation of approaches which sample vectors from a fixed orthogonal basis, the most efficient sampling method in terms of the number of random bits required in the literature [9], while the second strategy is novel and represents our main contribution. Both strategies have similar randomness requirements: In the first strategy at least $\lceil \log_2(n) \rceil$ random bits are necessary to ensure the possibility of choosing every element of B. In the second strategy, an identical number of random bits is necessary to choose $\mathbf{x}$ for a fixed B, and $\lceil \log_2(b) \rceil$ random bits are necessary to choose B. Note that an upper bound on the number of mutually unbiased bases is one greater than the dimensionality of the space, and this bound is saturated for spaces where the dimensionality is prime or an integer power of a prime, i.e. $b \leq n + 1$. Thus the number of random bits necessary to implement these strategies differs by a factor of approximately

two. The third and fourth strategies significantly outperform the fixed basis estimator in terms of single-shot variance, at the cost of a dramatic increase in the amount of randomness required, and have been extensively studied in the literature [9, 85, 144]. For conciseness we will not repeat the analysis of these methods in this paper but will compare the fixed basis estimator and MUBs estimator to them in Table 6.4.1.

6.3.1 Analysis of Fixed Basis Estimator

We first analyse the worst case variance of the fixed base estimator. In this analysis and the analysis for the MUBs estimator which follows, we make no assumption on A and consider the worst case variance.

We begin from the definition of the variance of the estimator for a single query. Let X be a random variable such that $X = \mathbf{x}^\dagger A \mathbf{x}$, where $\mathbf{x}$ is chosen according to the fixed basis strategy. Then

$$\text{Var(X)} = E(X^2) - E(X)^2,$$

where $E(\cdot)$ denotes the expectation value of the argument. We compute this term by term. First

$$E(X) = \frac{1}{n} \sum_{\mathbf{x} \in B} \mathbf{x}^\dagger A \mathbf{x} = \frac{\text{Tr}(A)}{n}.$$

where $n = \dim A$, and hence the second term in Eq. 6.3.1 is equal to $\frac{\text{Tr}(A)^2}{n^2}$. Turning to the first term,

$$\begin{aligned} E(X^2) &= \frac{1}{n} \sum_{\mathbf{x} \in B} \left(\mathbf{x}^\dagger A \mathbf{x} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} M_{ii}^2, \end{aligned}$$

where $M = UAU^\dagger$ for some fixed unitary matrix U such that $U^\dagger \mathbf{x}$ is a vector in the standard basis for all $x \in B$, and M_{ii} is the ith entry on the main diagonal of M. The variance for the fixed basis estimator is then given by $V_{\text{fixed}} = n\sum_{i=1}^{n} M_{ii}^2 - \text{Tr}(A)^2$. The worst case occurs when the value of $\sum_{i=1}^{n} M_{ii}^2$ is maximized for fixed trace of A (and hence M), and so the worst case single shot variance for the fixed basis estimator is $V_{fixed}^{\text{worst}} = (n-1)\text{Tr}(A)^2$.

6.3.2 Analysis of MUBS Estimator

We now turn to an analysis of the MUBs estimator. We assume that n is either prime or a prime raised to some integer power. In this case, it has been established that $b = n + 1$ [98]. The variance is defined as in Eq. 6.3.1, except that X is defined in terms of $\mathbf{x}$ chosen according to the MUBs strategy. Again, we analyse the individual terms making up the variance. We begin with

$$E(X) = \frac{1}{nb}\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\mathbf{x}^\dagger A\mathbf{x} = \frac{\text{Tr}(A)}{n}.$$

and hence the second term in the variance is the same as for the fixed basis estimator. Analysing the first term is, however, more difficult. We begin with the observation that $E(X^2)$ can be expressed in terms of the trace of the Kronecker product of two matrices, as follows

$$\begin{aligned} E(X^2) &= \frac{1}{nb}\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\left(\mathbf{x}^\dagger A\mathbf{x}\right)^2 \\ &= \frac{1}{nb}\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\text{Tr}\left((\mathbf{x}\mathbf{x}^\dagger A)^{\otimes 2}\right). \end{aligned}$$

Moving the summations inside the equation* we obtain

$$\begin{aligned} E(X^2) &= \frac{1}{nb}\text{Tr}\left(\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\left(\mathbf{x}\mathbf{x}^\dagger\right)^{\otimes 2}A^{\otimes 2}\right) \\ &= \frac{2}{nb}\text{Tr}\left(PA^{\otimes 2}\right), \end{aligned} \tag{6.1}$$

where $P = \frac{1}{2}\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\left(\mathbf{x}\mathbf{x}^\dagger\right)^{\otimes 2}$.

While this form of P may appear intimidating, we now prove that P is in fact a projector with each eigenvalue being either 0 or 1. We prove this indirectly, first by showing that P has rank at most $n(n+1)/2$, and then using the relationship between the traces of P and P^2 to conclude that the remaining $n(n+1)/2$ eigenvalues are equal to unity. Any vector of the form $\mathbf{w} = \mathbf{u}\otimes\mathbf{v} - \mathbf{u}\otimes\mathbf{v}$ for $\mathbf{u},\mathbf{v}\in B_1$ trivially satisfies $P\mathbf{w} = \mathbf{0}$. Since such vectors form a basis for a subspace of dimension $n(n-1)/2$, we conclude that $\text{rank}(P) \leq n^2 - n(n-1)/2 = n(n+1)/2$. Turning now to the issue of trace, we have

$$\begin{aligned}
\text{Tr}(P) &= \text{Tr}\left(\frac{1}{2}\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\left(\mathbf{x}\mathbf{x}^\dagger\right)^{\otimes 2}\right) \\
&= \frac{1}{2}\sum_{B\in\mathbb{B}}\sum_{\mathbf{x}\in B}\left(\mathbf{x}^\dagger\mathbf{x}\right)^2 \\
&= \frac{nb}{2}.
\end{aligned}$$

We can similarly compute the trace of P^2 to obtain

$$\begin{aligned}
\text{Tr}(P^2) &= \text{Tr}\left(\frac{1}{4}\sum_{B,B'\in\mathbb{B}}\sum_{\mathbf{x}\in B}\sum_{\mathbf{y}\in B'}\left(\mathbf{x}\mathbf{x}^\dagger\right)^{\otimes 2}\left(\mathbf{y}\mathbf{y}^\dagger\right)^{\otimes 2}\right) \\
&= \frac{1}{4}\sum_{B,B'\in\mathbb{B}}\sum_{\mathbf{x}\in B}\sum_{\mathbf{y}\in B'}\left|\mathbf{x}^\dagger\mathbf{y}\right|^4 \\
&= \frac{nb}{4} + \frac{n^2 b(b-1)}{4n^2} \\
&= \frac{b(n+b-1)}{4}.
\end{aligned}$$

Notice that this implies that $\text{Tr}(P) = \text{Tr}(P^2)$ for dimensions which are prime or integer powers of a prime, since in such cases $b = n+1$. This implies that the eigenvalues on the non-zero subspace minimize the sum of their squares for a fixed sum, and since P is positive semi-definite, we can conclude that each non-zero eigenvalue must be equal to unity.

Returning to the calculation of variance, we then have

$$\begin{aligned} E(X^2) &\leq \frac{2}{nb}\mathrm{Tr}\left(A^{\otimes 2}\right) \\ &= \frac{2}{nb}\mathrm{Tr}\,(A)^2, \end{aligned}$$

where $A^{\otimes 2}$ is the tensor power of 2 of A and hence

$$\mathrm{Var}(X) \leq \left(\frac{2}{nb} - \frac{1}{n^2}\right)\mathrm{Tr}\,(M)^2 \leq \frac{\mathrm{Tr}\,(A)^2}{n^2}.$$

This implies that the variance on the estimate of $\mathrm{Tr}(A)$ is bounded from above by $\mathrm{Tr}(A)^2$. It is, in fact, possible to compute the variance exactly from Eq. 6.1 by observing that M is the projector onto the symmetric subspace when n is an integer power of a prime. That is to say, for any vector $\mathbf{u}$ and any vector $\mathbf{v}$ orthogonal to $\mathbf{u}$, the vectors $\mathbf{u}\otimes\mathbf{v} + \mathbf{v}\otimes\mathbf{u}$, $\mathbf{u}\otimes\mathbf{u}$ and $\mathbf{v}\otimes\mathbf{v}$ are in the $+1$ eignespace of M, whereas the vector $\mathbf{u}\otimes\mathbf{v} - \mathbf{v}\otimes\mathbf{u}$ is in the null space of M. Thus we can compute the exact variance of the MUBs estimator, using the spectral decomposition $A = \sum_i \lambda_i \mathbf{u}_i\mathbf{u}_i^\dagger$ as

$$\begin{aligned} \mathrm{V}_{\mathrm{MUBs}} &= \frac{2n}{n+1}\mathrm{Tr}\left(PA^{\otimes 2}\right) - \mathrm{Tr}(A)^2 \\ &= \frac{2n}{n+1}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j \\ &\qquad \mathrm{Tr}\left(P(\mathbf{u}_i\otimes\mathbf{u}_j)(\mathbf{u}_i\otimes\mathbf{u}_j)^\dagger\right) - \mathrm{Tr}(A)^2 \\ &= \frac{2n}{n+1}\sum_{i=1}^{n}\left(\lambda_i^2 + \frac{1}{2}\sum_{j\neq i}\lambda_i\lambda_j\right) - \mathrm{Tr}(A)^2 \\ &= \frac{n}{n+1}\mathrm{Tr}(A^2) - \frac{1}{n+1}\mathrm{Tr}(A)^2. \end{aligned}$$

Since for all positive semi-definite matrices A the value of $\mathrm{Tr}(A)^2$ is bounded from below by $\mathrm{Tr}(A^2)$, the single shot variance on the MUBs estimator is bounded by $\mathrm{V}_{\mathrm{MUBs}}^{\mathrm{worst}} = \frac{n-1}{n+1}\mathrm{Tr}(A^2)$ in the worst case, a significant improvement on the bound

stemming from Eq. 6.3.2. The worst case single shot variance of the MUBs estimator is then at least a factor of $n+1$ better than that of any fixed basis estimator. Furthermore, the variance for the widely used Hutchinson estimator [85, 9], is given by $V_{\mathrm{H}} = 2\left(\mathrm{Tr}(A^2) - \sum_{i=1}^{n} A_{ii}^2\right)$. In the worst case, $\sum_{i=1}^{n} A_{ii}^2 = \frac{1}{n}\mathrm{Tr}(A^2)$, and hence the worst case single shot variance for Hutchinson estimator is $V_{\mathrm{H}}^{\mathrm{worst}} = \frac{2(n-1)}{n}\mathrm{Tr}(A^2)$. Thus, the MUBs estimator has better worst case performance than the Hutchinson estimator by a factor $\frac{2(n+1)}{n}$ which approaches 2 from above for large n.

6.4 Experimentation

6.4.1 Theoretical Resuls

Table 6.1 compares the single shot variance, worst case single shot variance and randomness requirements of the trace estimators. As can be seen from the comparison the MUBs estimator has strictly smaller variance than either the Hutchinson or Gaussian methods, while requiring significantly less randomness to implement. Given the drastic reduction in randomness requirements and the improved worst-case performance, the MUBs estimator provides an attractive alternative to previous methods for estimating the trace of implicit matrices.

6.4.2 Numerical Result

6.4.2.1 Example Matrix

Before we demonstrate the use of the MUBs estimator on example applications we draw the readers attention to a situation where the traditional methods perform poorly. This occurs when the values of the matrix A are close to the ones matrix with a small proportion of the diagonal values much greater. Due to the relationship between each of the unbiased bases this 'spikiness' only appears in one of the

Estimator	V	V^{worst}	R
Fixed basis	$n\sum_{i=1}^{n} M_{ii}^2 - \text{Tr}(A)^2$	$(n-1)\text{Tr}(A)^2$	$\log_2(n)$
MUBs	$\frac{n}{n+1}\text{Tr}(A^2) - \frac{1}{n+1}\text{Tr}(A)^2$	$\frac{n-1}{n+1}\text{Tr}(A^2)$	$\log_2(n) + \log_2(n+1)$
Hutchinson [85]	$2\left(\text{Tr}(A^2) - \sum_{i=1}^{n} A_{ii}^2\right)$	$\frac{2(n-1)}{n}\text{Tr}(A^2)$	n
Gaussian [144]	$2\text{Tr}(A^2)$	$2\text{Tr}(A^2)$	∞ for exact; $\mathcal{O}(n)$ for fixed precision

Table 6.1: Comparison of single shot variance V, worst case single shot variance V^{worst} and number of random bits R required for commonly used trace estimators and the MUBs estimator.

$n+1$ bases and hence the MUBs estimator appears very robust to the condition.

It is worth noting the reason we observe an order of magnitude improvement in this setting over the competing methods. The spikes matrix described can be written as the sum of two rank-one matrices. Each of these matrices will perform very poorly for the unitary estimator on that basis but gets exactly the correct result in the n other mutually unbiased bases. Naturally, as n becomes large and the number of samples utilised is relatively small, then we sample the exact result with high probability.

We can generalise this result to low-rank matrices more broadly. Any given rank-m matrix can be written as the sum of m rank-1 matrices. Figure 6.2, demonstrates the convergence of the stochastic trace estimators to rank-10 1000×1000 matrices. These were created by sampling 10 eigenvalues from a standard χ^2 distribution and sampling the first 10 eigenvectors of a Gaussian random matrix.

6.4.2.2 Counting Triangles in Graphs

As an example application, we will consider counting the number of triangles in a graph. This is an important problem in a number of application domains such

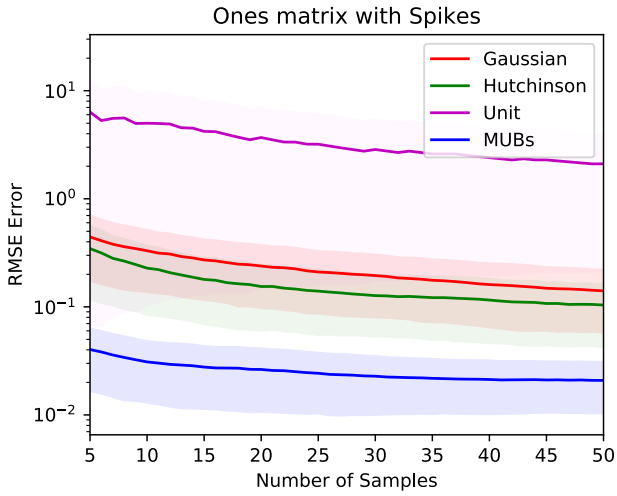


Figure 6.1: Convergence of the methods when estimating the trace of a 1000×1000 ones matrix with 1 diagonal of values replaced with 1001. This 'spike' has little effect of the convergence of the MUBs estimator and hence the method vastly outperforms the others. The experiment was run 500 times and the mean and standard deviation have been plotted for each method.

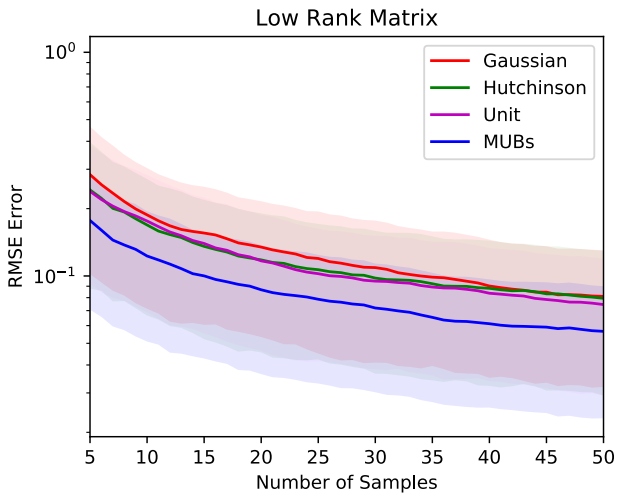


Figure 6.2: Convergence of the methods when estimating the trace of a 1000×1000 rank-10 matrix. The eigenvalues were sampled from a standard χ^2-distribution. As the rank of the matrix is only 1% of the dimensionality of the space we once again see substantially improved convergence rates. The experiment was run 30 times and the mean and standard deviation have been plotted for each method.

as identifying the number of 'friend of a friend' connections in a social network which is important for friendship suggestions [109, 159], identifying spam-like be-

haviour [17] and even identifying thematic structures on the internet [53]. An efficient method to do this is the Trace Triangle algorithm [8]. The algorithm is based on a relationship between the adjacency matrix, A, and the number of triangles for an undirected graph, Δ_g,

$$\Delta_g = \frac{Tr(A^3)}{6}.$$

The trace of the adjacency matrix cubed can be sampled in $\mathcal{O}(n^2)$ per sample as opposed to being explicitly computed in $\mathcal{O}(n^3)$. We compared Gaussian, Hutchinson's, Unit and MUBs estimators performance at predicting the number of triangles for the graphs presented in Table 6.2 and the results of the experiment are presented in Figure 6.3. The code for these experiments, with an efficient Python implementation for generating the MUBs sample vectors in $\mathcal{O}(n)$, is available at `github.com/OxfordML/MUBs_TraceEstimators`. The MUBs estimator outperforms each of the classical methods in all of the experiments, as would be implied by the theory.

Dataset	Vertices	Edges	Triangles
Arxiv-HEP-th	27,240	341,923	1,478,735
CA-AstroPh	18,772	198,050	1,351,441
CA-GrQc	5,242	14,484	48,260
wiki-vote	7,115	100,689	608,389

Table 6.2: Datasets used for the comparison of stochastic trace estimation methods in the counting of triangles in graphs. All datasets can be found at `snap.stanford.edu/data`

6.4.2.3 Log Determinant of Covariance Matrix

Next, let us consider a common linear algebraic calculation required in the training of Gaussian processes, determinantal point processes and Gauss Markov random

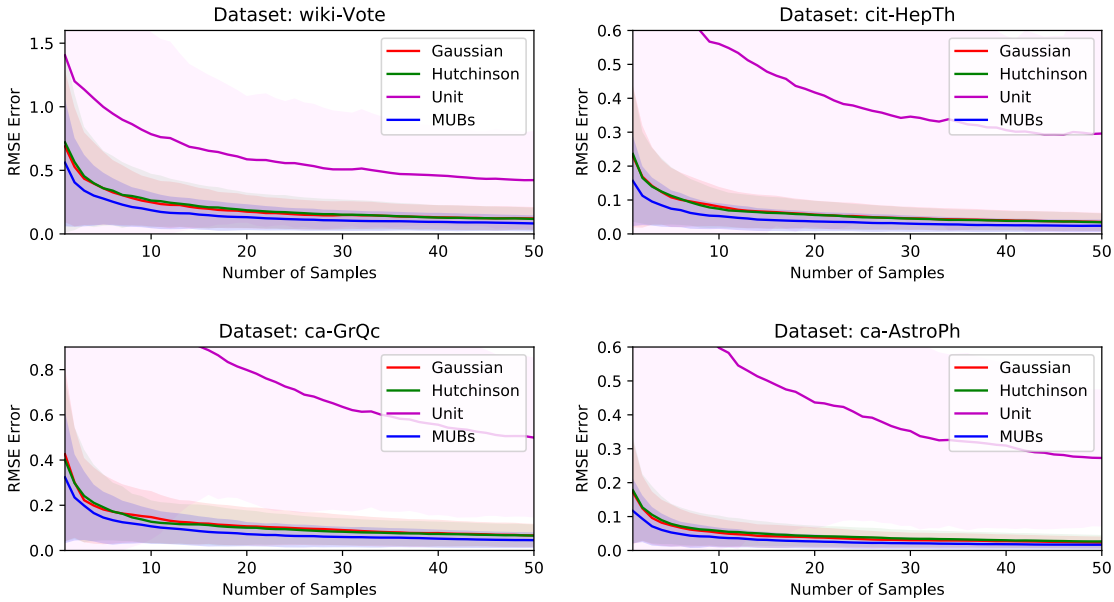


Figure 6.3: A comparison of the performance of the stochastic trace estimation methods on the four datasets. The fixed basis method was not included as it was not competitive. The experiments were performed 500 times each. The solid line indicated the empirical mean absolute relative error and the surrounding transparent region indicates one empirical standard deviation of the 500 trials.

field modelling to name just a few applications, namely the log determinant of a kernel matrix.

The use of stochastic trace estimation to approximate log determinant calculations of kernel matrices has been well studied [77, 56, 58] and a range of methods are feasible. Most notably, polynomial approaches such as truncated Taylor approximations and Chebyshev approximations [77, 120] have been applied, with the latter achieving consistently better results. The general concept relies on the fact that the trace of a matrix is simply the sum of its eigenvalues and the log-determinant is the sum of the log of its eigenvalues. Stochastic trace estimation aids us in approximating the sum of the eigenvalues squared, cubed and so on which we can use in a polynomial approximation of the log function,

$$\log(x) \approx \sum_{j=0}^{m} c_j x^j \quad \rightarrow \quad \log(|K|) \approx \sum_{j=0}^{m} c_j \mathrm{Tr}(K^j)$$

where the constants c_j refer to the coefficients of the polynomial approximation. In practice, the trace of K^0 is simply the dimensionality of the matrix, K^1 is the

trace of the explicit matrix and K^2 can be found as $\sum_{i,j} K_{i,j}$ due to the relationship between the matrix elements and the Frobenius norm. As such, the approximation error is only incurred is only due to the trace of the matrix raised to three and above.

In order to demonstrate the effect of improved stochastic trace estimation on log determinant estimations, we sampled 1000 points from a 5-dimensional hypercube uniformly at random. These points, in turn, formed a covariance matrix using an isotropic Gaussian kernel function. This aimed to emulate a realistic dataset which may be used by practitioners.

We used an order-6 Chebyshev polynomial approximation and recorded estimation errors of the relative root mean squared error (RMSE) for each power of the covariance matrix. These can be seen in Figure 6.4. Also plotted is the estimation error of the log-determinant itself, as it compounds both the polynomial approximation error and the error due to the stochastic trace estimation. A fixed budget of 25 probing vectors was allowed for each of the approaches. As can be seen in the figure, the error incurred due to the stochastic trace estimation is non-negligible and for the higher order estimates the MUBs approach was achieving improved results in turns of both its expectation and standard error.

6.5 Conclusion

Sparse matrices offer the opportunity for us to leverage computationally cheap stochastic approaches to linear algebraic operations. Stochastic trace estimation is a prime example of this. We have introduced a new MUBs sampler for stochastic trace estimation which combines the efficiency of fixed basis methods with a performance which outperforms the state of the art methods. We offer both empir-

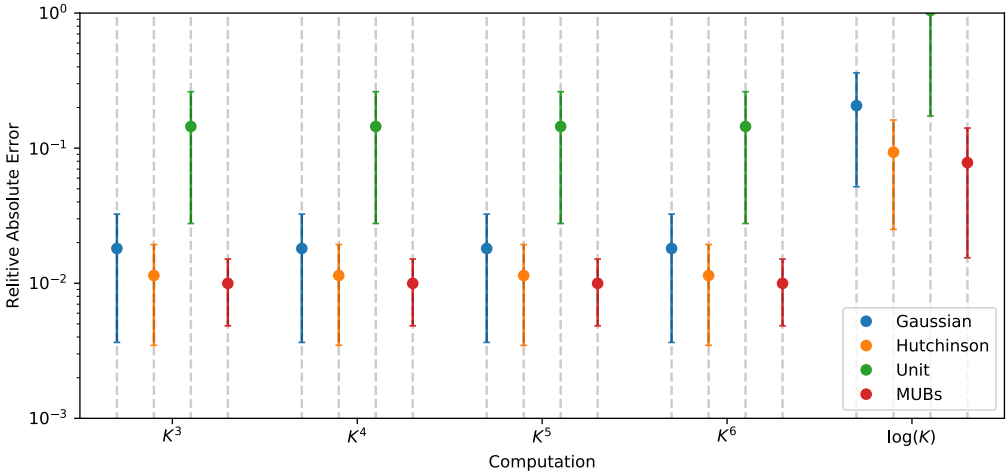


Figure 6.4: The performance of estimating the trace of K^3, K^4, K^5, K^6 and their combined result in the Chebyshev polynomial approximation of $\log(|K|)$. The experiment we ran 20 times and their expectation and standard error have been shown above.

ical and theoretical comparisons to the previously established state of the art techniques and clearly demonstrate the benefit of using mutually unbiased bases for stochastic linear algebraic procedures to accelerate machine learning algorithms.

All code used in these experiments has been made available at `github.com/OxfordML/MUBs_Trac`

Chapter 7

Probabilistic Numeric Eigen-Spectrum Inference

The previous chapter introduced stochastic trace estimation which can be used to estimate the trace of implicit matrices, such as matrix powers without exact computation. This is a powerful tool which is exploited in this chapter in order to make Bayesian estimates of log determinant calculations.

7.1 Introduction

Developing scalable learning models without compromising performance is at the forefront of machine learning research. The scalability of several learning models is predominantly hindered by linear algebraic operations having large computational complexity, among which is the computation of the log-determinant of a matrix [69]. The latter term features heavily in the machine learning literature, with applications including spatial models [6, 135], kernel-based models [41, 130], and Bayesian learning [107].

The standard approach for evaluating the log-determinant of a positive definite matrix involves the use of Cholesky decomposition [69], which is employed in various applications of statistical models such as kernel machines. However, the use

of Cholesky decomposition for general dense matrices requires $\mathcal{O}(n^3)$ operations, whilst also entailing memory requirements of $\mathcal{O}(n^2)$. In view of this computational bottleneck, various models requiring the log-determinant for inference bypass the need to compute it altogether [2, 149, 39, 55].

Alternatively, several methods exploit sparsity and structure within the matrix itself to accelerate computations. For example, sparsity in Gaussian Markov Random fields (GMRFs) arises from encoding conditional independence assumptions that are readily available when considering low-dimensional problems. For such matrices, the Cholesky decompositions can be computed in fewer than $\mathcal{O}(n^3)$ operations [135, 137]. Similarly, Kronecker-based linear algebra techniques may be employed for kernel matrices computed on regularly spaced inputs [138]. While these ideas have proven successful for a variety of specific applications, they cannot be extended to the case of general dense matrices without assuming special forms or structures for the available data.

To this end, general approximations to the log-determinant frequently build upon stochastic trace estimation techniques using iterative methods [10]. Two of the most widely-used polynomial approximations for large-scale matrices are the Taylor and Chebyshev expansions [6, 78]. A more recent approach draws from the possibility of estimating the trace of functions using stochastic Lanczos quadrature [153], which has been shown to outperform polynomial approximations from both a theoretic and empirical perspective.

Inspired by recent developments in the field of probabilistic numerics [83], in this work we propose an alternative approach for calculating the log-determinant of a matrix by expressing this computation as a Bayesian quadrature problem. In doing so, we reformulate the problem of *computing* an intractable quantity into

an *estimation* problem, where the goal is to infer the correct result using tractable computations that can be carried out within a given time budget. In particular, we model the eigenvalues of a matrix A from noisy observations of $\mathrm{Tr}(A^k)$ obtained through stochastic trace estimation using the Taylor approximation method [169]. Such a model can then be used to make predictions on the infinite series of the Taylor expansion, yielding the estimated value of the log-determinant. Aside from permitting a probabilistic approach for predicting the log-determinant, this approach inherently yields uncertainty estimates for the predicted value, which in turn serves as an indicator of the quality of our approximation.

The contribution of this chapter are as follows.

1. We propose a probabilistic approach for computing the log-determinant of a matrix which blends different elements from the literature on estimating log-determinants under a Bayesian framework.

2. We demonstrate how bounds on the expected value of the log-determinant improve our estimates by constraining the probability distribution to lie between designated lower and upper bounds.

3. Through rigorous numerical experiments on synthetic and real data, we demonstrate how our method can yield superior approximations to competing approaches, while also having the additional benefit of uncertainty quantification.

4. Finally, in order to demonstrate how this technique may be useful within a practical scenario, we employ our method to carry out parameter selection for a large-scale determinantal point process.

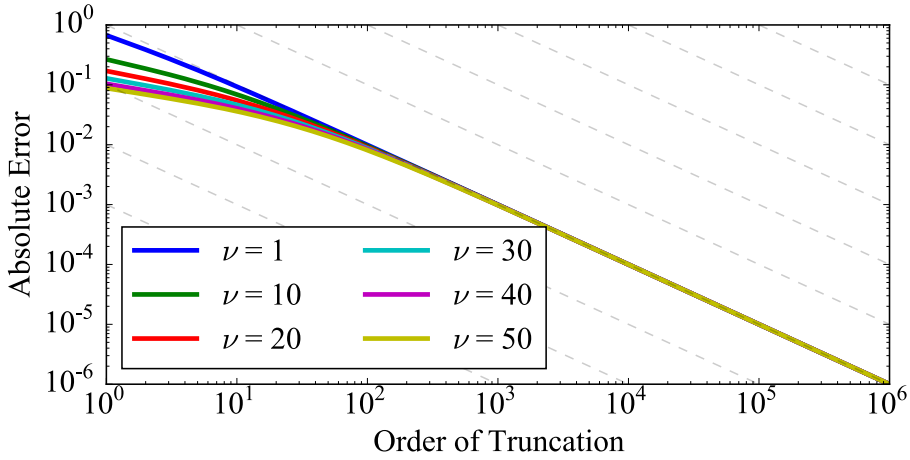


Figure 7.1: Expected absolute error of truncated Taylor series for stationary ν-continuous kernel matrices. The dashed grey lines indicate $\mathcal{O}(n^{-1})$.

To the best of our knowledge, this is the first time that the approximation of log-determinants is viewed as a Bayesian inference problem, with the resulting quantification of uncertainty being hitherto unexplored thus far.

7.2 Background

As highlighted in the introduction, several approaches for approximating the log-determinant of a matrix rely on stochastic trace estimation for accelerating computations. This comes about as a result of the relationship between the log-determinant of a matrix, and the corresponding trace of the log-matrix, whereby

$$\log\big(\mathrm{Det}\,(A)\big) = \mathrm{Tr}\big(\log\,(A)\big). \tag{7.1}$$

Provided the matrix $\log(A)$ can be efficiently sampled, this simple identity enables the use of stochastic trace estimation techniques [10, 59]. We elaborate further on this concept below.

7.2.1 Taylor Approximation

Against the backdrop of machine learning applications, in this work we predominantly consider covariance matrices taking the form of a Gram matrix $K = \left\{\kappa(\mathbf{x}_i, \mathbf{x}_j)\right\}_{i,j=1,\ldots,n}$, where the kernel function κ implicitly induces a feature space representation of data points $\mathbf{x}_i$. Assume K has been normalized such that the maximum eigenvalue is less than or equal to one, $\lambda_0 \leq 1$, where the largest eigenvalue can be efficiently found using Gershgorin intervals [64]. Given that covariance matrices are positive semidefinite, we also know that the smallest eigenvalue is bounded by zero, $\lambda_n \geq 0$. Motivated by the identity presented in (7.1), the Taylor series expansion [16, 169] may be employed for evaluating the log-determinant of matrices having eigenvalues bounded between zero and one. In particular, this approach relies on the following logarithm identity,

$$\log(I - A) = -\sum_{k=1}^{\infty} \frac{A^k}{k}. \tag{7.2}$$

While the infinite summation is not explicitly computable in finite time, this may be approximated by computing a truncated series instead. Furthermore, given that the trace of matrices is additive, we find

$$\mathrm{Tr}\big(\log(I - A)\big) \approx -\sum_{k=1}^{m} \frac{\mathrm{Tr}\left(A^k\right)}{k}. \tag{7.3}$$

The $\mathrm{Tr}(A^k)$ term can be computed efficiently and recursively by propagating $\mathcal{O}(n^2)$ vector-matrix multiplications in a stochastic trace estimation scheme. To compute $\mathrm{Tr}(\log(K))$ we simply set $A = I - K$.

There are two sources of error associated with this approach; the first due to stochastic trace estimation, and the second due to truncation of the Taylor series. In the case of covariance matrices, the smallest eigenvalue tends to be very small,

which can be verified by [162] and [145]'s observations on the eigenspectra of covariance matrices. This leads to A^k decaying slowly as $k \to \infty$.

In light of the above, standard Taylor approximations to the log-determinant of covariance matrices are typically unreliable, even when the exact traces of matrix powers are available. This can be verified analytically based on results from kernel theory, which state that the approximate rate of decay for the eigenvalues of positive definite kernels which are ν-continuous is $\mathcal{O}(n^{-\nu-0.5})$ [162, 160]. Combining this result with the absolute error, $E(\lambda)$, of the truncated Taylor approximation we find

$$\begin{aligned}
\mathbf{E}\left[E\left(\lambda\right)\right] &= \mathcal{O}\left(\int_0^1 \lambda^{\nu+0.5}\left(\log\left(\lambda\right) - \sum_{j=1}^{m}\frac{\lambda^j}{j}\right)\mathrm{d}\lambda\right) \\
&= \mathcal{O}\left(\int_0^1 \lambda^{\nu+0.5}\sum_{j=m}^{\infty}\frac{\lambda^j}{j}\mathrm{d}\lambda\right) \\
&= \mathcal{O}\left(\frac{\Psi^{(0)}\left(m+\nu+1.5\right) - \Psi^{(0)}\left(m\right)}{\nu+1.5}\right),
\end{aligned}$$

where $\Psi^{(0)}(\cdot)$ is the Digamma function. In Figure 7.1, we plot the relationship between the order of the Taylor approximation and the expected absolute error. It can be observed that irrespective of the continuity of the kernel, the error converges at a rate of $\mathcal{O}(n^{-1})$.

7.3 A Probabilistic Numerics Approach

We now propose a probabilistic numerics [83] approach: we'll re-frame a numerical computation (in this case, trace estimation) as probabilistic inference. Probabilistic numerics usually requires distinguishing: an appropriate latent function; data and; the ultimate object of interest. Given the data, a posterior distribution is calculated for the object of interest. For instance, in numerical integration, the

latent function is the integrand, f, the data are evaluations of the integrand, $f(x)$, and the object of interest is the value of the integral, $\int f(x)p(x)\mathrm{d}x$ (see § 7.3.1.1 for more details). In this work, our latent function is the distribution of eigenvalues of A, the data are noisy observations of $\mathrm{Tr}(A^k)$, and the object of interest is $\log(\mathrm{Det}(K))$. For this object of interest, we are able to provide both expected value and variance. That is, although the Taylor approximation to the log-determinant may be considered unsatisfactory, the intermediate trace terms obtained when raising the matrix to higher powers may prove to be informative if considered as observations within a probabilistic model.

7.3.1 Raw Moment Observations

We wish to model the eigenvalues of A from noisy observations of $\mathrm{Tr}\left(A^k\right)$ obtained through stochastic trace estimation, with the ultimate goal of making predictions on the infinite series of the Taylor expansion. Let us assume that the eigenvalues are i.i.d. random variables drawn from $P(\lambda_i = x)$, a probability distribution over $x \in [0,1]$. In this setting $\mathrm{Tr}(A) = n\mathbf{E}_x[P(\lambda_i = x)]$, and more generally $\mathrm{Tr}\left(A^k\right) = n\mathbf{R}_x^{(k)}[P(\lambda_i = x)]$, where $\mathbf{R}_x^{(k)}$ is the k^{th} raw moment over the x domain. The raw moments can thus be computed as,

$$\mathbf{R}_x^{(k)}\left[P\left(\lambda_i = x\right)\right] = \int_0^1 x^k P\left(\lambda_i = x\right)\mathrm{d}x. \tag{7.4}$$

Such a formulation is appealing because if $P\left(\lambda_i = x\right)$ is modelled as a Gaussian process, the required integrals may be solved analytically using Bayesian Quadrature.

7.3.1.1 Bayesian Quadrature

Gaussian processes [130] are a powerful Bayesian inference method defined over functions $X \to \mathbb{R}$, such that the distribution of functions over any finite subset of the input points $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is a multivariate Gaussian distribution. Under this framework, the moments of the conditional Gaussian distribution for a set of predictive points, given a set of labels $\mathbf{y} = (y_1, \ldots, y_n)^\top$, may be computed as

$$\mu = \mu_0 + K_*^\top K^{-1}(\mathbf{y} - \mu_0), \tag{7.5}$$

$$\Sigma = K_{*,*} - K_*^\top K^{-1} K_*, \tag{7.6}$$

with μ and Σ denoting the posterior mean and variance, and K being the $n \times n$ covariance matrix for the observed variables $\{\mathbf{x}_i, y_i; i \in (1, 2, \ldots n)\}$. The latter is computed as $\kappa(\mathbf{x}, \mathbf{x}')$ for any pair of points $\mathbf{x}, \mathbf{x}' \in X$. Meanwhile, K_* and $K_{*,*}$ respectively denote the covariance between the observable and the predictive points, and the prior over the predicted points. Note that μ_0, the prior mean, may be set to zero without loss of generality.

Bayesian Quadrature [117] is primarily concerned with performing integration of potentially intractable functions. In this work, we limit our discussion to the setting where the integrand is modeled as a GP,

$$\int p(x) f(x) \, \mathrm{d}x, \quad f \sim \mathrm{GP}(\mu, \Sigma),$$

where $p(x)$ is some measure with respect to which we are integrating. A full discussion of BQ may be found in [117] and [132]; for the sake of conciseness, we only state the result that the integrals may be computed by integrating the covariance function with respect to $p(x)$ for both K_*,

$$\kappa\left(\int x \mathrm{d}x, x'\right) = \int p\left(x\right) \kappa\left(x, x'\right) \mathrm{d}x,$$

and $K_{*,*}$,

$$\kappa\left(\int x\mathrm{d}x, \int x'\mathrm{d}x'\right) = \iint p\left(x\right)\kappa\left(x,x'\right)p\left(x'\right)\mathrm{d}x\mathrm{d}x'.$$

7.3.2 Inference on the Log-Determinant

Recalling (7.4), if $P(\lambda_i = x)$ is modeled using a GP, in order to include observations of $\mathbf{R}_x^{(k)}[P(\lambda_i = x)]$, denoted as $\mathbf{R}_x^{(k)}$, we must be able to integrate the kernel with respect to the polynomial in x,

$$\kappa\left(\mathbf{R}_x^{(k)}, x'\right) = \int_0^1 x^k \kappa\left(x,x'\right)\mathrm{d}x, \tag{7.7}$$

$$\kappa\left(\mathbf{R}_x^{(k)}, \mathbf{R}_{x'}^{(k')}\right) = \int_0^1\int_0^1 x^k \kappa\left(x,x'\right)x'^{k'}\mathrm{d}x\mathrm{d}x'. \tag{7.8}$$

Although the integrals described above are typically analytically intractable, certain kernels have an elegant analytic form which allows for efficient computation. In this section, we derive the raw moment observations for a histogram kernel and demonstrate how estimates of the log-determinant can be obtained.

7.3.2.1 Histogram Kernel

The entries of the histogram kernel, also known as the piecewise constant kernel, are given by $\kappa(x,x') = \sum_{j=0}^{1-m}\mathcal{H}(\frac{j}{m}, \frac{j+1}{m}, x, x')$, where

$$\mathcal{H}\left(\frac{j}{m}, \frac{j+1}{m}, x, x'\right) = \begin{cases} 1 & x,x' \in \left[\frac{j}{m}, \frac{j+1}{m}\right] \\ 0 & \text{otherwise} \end{cases}.$$

Covariances between raw moments may be computed as follows:

$$\begin{aligned}\kappa\left(\mathbf{R}_x^{(k)}, x'\right) &= \int_0^1 x^k\kappa\left(x,x'\right)\mathrm{d}x \\ &= \frac{1}{k+1}\left(\left(\frac{j+1}{m}\right)^{k+1} - \left(\frac{j}{m}\right)^{k+1}\right),\end{aligned} \tag{7.9}$$

where in the above x lies in the interval $\left[\frac{j}{m}, \frac{j+1}{m}\right]$. Extending this to the covariance function between raw moments we have,

$$\begin{aligned}\kappa\left(\mathbf{R}_x^{(k)}, \mathbf{R}_{x'}^{(k')}\right) &= \int_0^1 \int_0^1 x^k, x'^{k'} \kappa\left(x, x'\right) \mathrm{d}x \mathrm{d}x' \\ &= \sum_{j=0}^{m-1} \prod_{\bar{k} \in (k,k')} \frac{1}{(\bar{k}+1)} \left(\left(\frac{j+1}{m}\right)^{\bar{k}+1} - \left(\frac{j}{m}\right)^{\bar{k}+1}\right).\end{aligned} \tag{7.10}$$

This simple kernel formulation between observations of the raw moments compactly allows us to perform inference over $P(\lambda_i = x)$. However, the ultimate goal is to predict $\log(\mathrm{Det}(K))$, and hence $\sum_{i=1}^{\infty} \frac{\mathrm{Tr}\left(A^k\right)}{k}$. This requires a seemingly more complex set of kernel expressions; nevertheless, by propagating the implied infinite summations into the kernel function, we can also obtain the closed form solutions for these terms,

$$\begin{aligned}\kappa\left(\sum_{k=1}^{\infty} \frac{\mathbf{R}_x^{(k)}}{k}, \mathbf{R}_{x'}^{(k')}\right) = \sum_{j=0}^{m-1} \frac{1}{k'+1} \Bigg(\Bigg(\frac{j+1}{m}\Bigg)^{k+1} - \\ \left(\frac{j}{m}\right)^{k+1}\Bigg)\left(S\left(\frac{j+1}{m}\right) - S\left(\frac{j}{m}\right)\right)\end{aligned} \tag{7.11}$$

$$\kappa\left(\sum_{k=1}^{\infty} \frac{\mathbf{R}_x^{(k)}}{k}, \sum_{k'=1}^{\infty} \frac{\mathbf{R}_{x'}^{(k')}}{k'}\right) = \sum_{j=0}^{m-1} \left(S\left(\frac{j+1}{m}\right) - S\left(\frac{j}{m}\right)\right)^2 \tag{7.12}$$

where $S(\alpha) = \sum_{k=1}^{\infty} \frac{\alpha^{k+1}}{k(k+1)}$, which has the convenient identity for $0 < \alpha < 1$,

$$S(\alpha) = \alpha + (1-\alpha)\log(1-\alpha).$$

Following the derivations presented above, we can finally go about computing the prediction for the log-determinant, and its corresponding variance, using the GP posterior equations given in (7.5) and (7.6). This can be achieved by replacing the terms K_* and $K_{*,*}$ with the constructions presented in (7.11) and (7.12), respectively. The entries of K are filled in using (7.10), whereas $\mathbf{y}$ denotes the noisy observations of $\mathrm{Tr}\left(A^k\right)$.

7.3.2.2 Polynomial Kernel

Similar to the derivation of the histogram kernel, we can also derive the polynomial kernel for moment observations. The entries of the polynomial kernel, given by $k(x, x') = (xx' + c)^d$, can be integrated over as,

$$\begin{aligned}\kappa\left(\mathbf{R}_x^{(k)}, x'\right) &= \int_0^1 \sum_{i=1}^{d} \binom{d}{i} x^{k+i} x'^i c^{d-i} dx, \\ &= \sum_{i=1}^{d} \binom{d}{i} \frac{x'^i c^{d-i}}{k+i+1}.\end{aligned} \tag{7.13}$$

$$\begin{aligned}\kappa\left(\mathbf{R}_x^{(k)}, \mathbf{R}_{x'}^{(k')}\right) &= \int_0^1 \int_0^1 \sum_{i=1}^{d} \binom{d}{i} x^{k+i} x'^{k'+i} c^{d-i} dx dx' \\ &= \sum_{i=1}^{d} \binom{d}{i} \frac{c^{d-i}}{(k+i+1)(k'+i+1)}.\end{aligned} \tag{7.14}$$

As with the histogram kernel, the infinite sum of the Taylor expansion can also be combined into the Gaussian process,

$$\begin{aligned}\kappa\left(\sum_{k=1}^{\infty} \frac{\mathbf{R}_x^{(k)}}{k}, \mathbf{R}_{x'}^{(k')}\right) &= \frac{1}{k} \sum_{k=1}^{\infty} \sum_{i=1}^{d} \binom{d}{i} \frac{c^{d-i}}{(k+i+1)(k'+i+1)} \\ &= \sum_{i=1}^{d} \binom{d}{i} \frac{c^{d-i}\left(\Psi^{(0)}(i+2) + \gamma\right)}{(i+1)(k'+i+1)},\end{aligned} \tag{7.15}$$

$$\begin{aligned}\kappa\left(\sum_{k=1}^{\infty} \frac{\mathbf{R}_x^{(k)}}{k}, \sum_{k'=1}^{\infty} \frac{\mathbf{R}_{x'}^{(k')}}{k'}\right) &= \frac{1}{kk'} \sum_{k=1}^{\infty} \sum_{k'=1}^{\infty} \sum_{i=1}^{d} \binom{d}{i} \frac{c^{d-i}}{(k+i+1)(k'+i+1)} \\ &= \sum_{i=1}^{d} \binom{d}{i} \frac{c^{d-i}\left(\Psi^{(0)}(i+2) + \gamma\right)^2}{(i+1)^2}.\end{aligned} \tag{7.16}$$

In the above, $\Psi^{(0)}(\cdot)$ is the Digamma function and γ is the Euler-Mascheroni constant. We strongly believe that the polynomial and histogram kernels are not the only kernels which can analytically be derived to include moment observations but act as a reasonable initial choice for practitioners.

7.3.2.3 Prior Mean Function

While GPs, and in this case BQ, can be applied with a zero mean prior without loss of generality, it is often beneficial to have a mean function as an initial starting point. If $P(\lambda_i = x)$ is composed of a constant mean function $g(\lambda_i = x)$, and a GP is used to model the residual, we have that

$$P\left(\lambda_i = x\right) = g\left(\lambda_i = x\right) + f\left(\lambda_i = x\right).$$

The previously derived moment observations may then be decomposed into,

$$\begin{aligned}\int x^k P\left(\lambda_i = x\right) \mathrm{d}x = &\int x^k g\left(\lambda_i = x\right) \mathrm{d}x \\ &+ \int x^k f\left(\lambda_i = x\right) \mathrm{d}x.\end{aligned} \tag{7.17}$$

Due to the domain of $P\left(\lambda_i = x\right)$ lying between zero and one, we set a Beta distribution as the prior mean, which has some convenient properties. First, it is fully specified by the mean and variance of the distribution, which can be computed using the trace and Frobenius norm of the matrix. Secondly, the r-th raw moment of a Beta distribution parameterized by α and β is

$$\mathbf{R}_x^{(k)}\left[g\left(\lambda_i = x\right)\right] = \frac{\alpha + r}{\alpha + \beta + r},$$

which is straightforward to compute.

In consequence, the expectation of the logarithm of random variables and, hence, the 'prior' log determinant yielded by $g\left(\lambda_i = x\right)$ can be computed as

$$\mathbf{E}[\log(X); X \sim g(\lambda_i = x)] = \phi(\alpha) - \phi(\alpha + \beta). \tag{7.18}$$

This can then simply be added to the previously derived GP expectation of the log-determinant.

7.3.2.4 Bai & Golub Log-Determinant Bounds

For the sake of completeness, we restate the bounds on the log-determinants used throughout this chapter [12].

Let A be an n-by-n symmetric positive definite matrix, $\mu_1 = \mathrm{Tr}(A)$, $\mu_2 = \|A\|_F^2$ and $\lambda_i(A) \in [\alpha;\beta]$ with $\alpha > 0$, then

$$\begin{bmatrix} \log\alpha \\ \log t \end{bmatrix}^T \begin{bmatrix} \alpha & t \\ \alpha^2 & t^2 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \leq \mathrm{Tr}(\log(A)) \leq \begin{bmatrix} \log\beta \\ \log\bar{t} \end{bmatrix}^T \begin{bmatrix} \beta & \bar{t} \\ \beta^2 & \bar{t}^2 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

where,

$$t = \frac{\alpha\mu_1 - \mu_2}{\alpha n - \mu_2}, \quad \bar{t} = \frac{\beta\mu_1 - \mu_2}{\beta n - \mu_2}$$

This bound can be easily computed during the loading of the matrix as both the trace and Frobenius norm can be readily calculated using summary statistics. However, bounds on the maximum and minimum must also be derived. We chose to use Gershgorin intervals to bound the eigenvalues [64].

7.3.2.5 Using Bounds on the Log-Determinant

As with most GP specifications, there are hyperparameters associated with the prior and the kernel. The optimal settings for these parameters may be obtained via optimization of the standard GP log marginal likelihood, defined as

$$\mathrm{LML}_{GP} = -\frac{1}{2}\mathbf{y}^\top K^{-1}\mathbf{y} - \frac{1}{2}\log(\mathrm{Det}(K)) + \text{const.}$$

Borrowing from the literature on bounds for the log-determinant of a matrix, as described in section 7.3.2.4, we can also exploit such upper and lower bounds to

truncate the resulting GP distribution to the relevant domain, which is expected to greatly improve the predicted log-determinant. These additional constraints can then be propagated to the hyperparameter optimization procedure by incorporating them into the likelihood function via the product rule, as follows:

$$\text{LML} = \text{LML}_{GP} + \log\left(\Phi\left(\frac{a-\hat{\mu}}{\hat{\sigma}}\right) - \Phi\left(\frac{b-\hat{\mu}}{\hat{\sigma}}\right)\right),$$

with a and b representing the upper and lower log-determinant bounds respectively, $\hat{\mu}$ and $\hat{\sigma}$ representing the posterior mean and standard deviation, and $\Phi(\cdot)$ representing the Gaussian cumulative density function. Priors on the hyperparameters may be accounted for in a similar way.

7.3.2.6 Algorithm Complexity and Recap

Due to its cubic complexity, GP inference is typically considered detrimental to the scalability of a model. However, in our formulation, the GP is only being applied to the noisy observations of $\text{Tr}\left(A^k\right)$, which rarely exceed the order of tens of points. As a result, given that we assume this to be orders of magnitude smaller than the dimensionality n of the matrix K, the computational complexity is dominated by the matrix-vector operations involved in stochastic trace estimation, i.e. $\mathcal{O}(n^2)$ for dense matrices and $\mathcal{O}(ns)$ for s-sparse matrices.

The steps involved in the procedure described in this section are summarized as pseudo-code in Algorithm 1. The input matrix A is first normalized by using Gershgorin intervals to find the largest eigenvalue (line 1), and the expected bounds on the log-determinant (line 2) are calculated using matrix theory (Appendix 7.3.2.4). The noisy Taylor observations up to an expansion order M (lines 3-4), denoted here as $\mathbf{y}$, are then obtained through stochastic trace estimation, as described in § 7.2.1. These can be modelled using a GP, where the entries of the

kernel matrix K (lines 5-7) are computed using (7.10). The kernel parameters are then tuned as per § 7.3.2.5 (line 8). Recall that we seek to make a prediction for the infinite Taylor expansion, and hence the exact log-determinant. To this end, we must compute K_* (lines 9-10) and $k_{*,*}$ (line 11) using (7.11) and (7.12), respectively. The posterior mean and variance (line 12) may then be evaluated by filling in (7.5) and (7.6). As outlined in the previous section, the resulting posterior distribution can be truncated using the derived bounds to obtain the final estimates for the log-determinant and its uncertainty (line 13).

Algorithm 1 Computing log-determinant and uncertainty using probabilistic numerics

Input: PSD matrix $A \in \mathbb{R}^{n \times n}$, raw moments kernel κ, expansion order M, and random vectors Z

Output: Posterior mean MTRN, and uncertainty VTRN

1: $A \leftarrow \text{NORMALIZE}(A)$
2: $\text{BOUNDS} \leftarrow \text{GETBOUNDS}(A)$
3: **for** $i \leftarrow 1$ to M **do**
4: $\quad \mathbf{y}_i \leftarrow \text{STOCHASTICTAYLOROBS}(A, i, \text{Z})$
5: **for** $i \leftarrow 1$ to M **do**
6: $\quad$ **for** $j \leftarrow 1$ to M **do**
7: $\quad\quad K_{ij} \leftarrow \kappa(i, j)$
8: $\kappa, K \leftarrow \text{TUNEKERNEL}(K, \mathbf{y}, \text{BOUNDS})$
9: **for** $i \leftarrow 1$ to M **do**
10: $\quad K_{*,i} \leftarrow \kappa(*, i)$
11: $k_{*,*} \leftarrow \kappa(*, *)$
12: $\text{MEXP}, \text{VEXP} \leftarrow \text{GPPRED}(\mathbf{y}, K, K_*, k_{*,*})$
13: $\text{MTRN}, \text{VTRN} \leftarrow \text{TRUNC}(\text{MEXP}, \text{VEXP}, \text{BOUNDS})$

7.4 Experiments

In this section, we show how the appeal of this formulation extends beyond its intrinsic novelty, whereby we also consistently obtain performance improvements over competing techniques. We set up a variety of experiments for assessing the

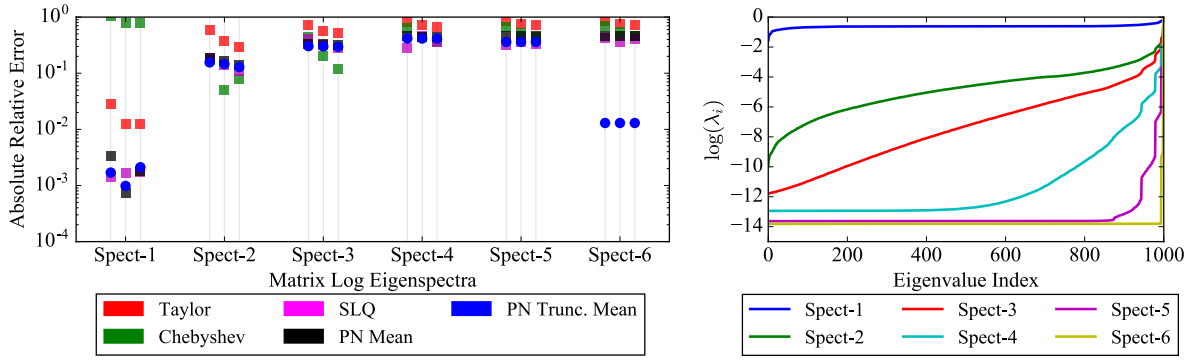


Figure 7.2: Empirical performance of 6 covariances described in § 7.4.1. The right figure displays the log eigenspectrum of the matrices and their respective indices. The left figure displays the relative performance of the algorithms for the stochastic trace estimation order set to 5, 25 and 50 (from left to right respectively).

model performance, including both synthetically constructed and real matrices. Given the model's probabilistic formulation, we also assess the quality of the uncertainty estimates yielded by the model. We conclude by demonstrating how this approach may be fitted within a practical learning scenario.

We compare our approach against several other estimations to the log-determinant, namely approximations based on Taylor expansions, Chebyshev expansions and Stochastic Lanczos quadrature. The Taylor approximation has already been introduced in § 7.2.1, and we briefly describe the others below.

Chebyshev Expansions: This approach utilizes the m-degree Chebyshev polynomial approximation to the function $\log(I - A)$ [78, 24, 123],

$$\mathrm{Tr}\left(\log\left(I - A\right)\right) \approx \sum_{k=0}^{m} c_k \mathrm{Tr}\left(T_k\left(A\right)\right), \tag{7.19}$$

where $T_k(x) = AT_{k-1}(A) - T_{k-2}(A)$ starting with $T_0(A) = 1$ and $T_0(A) = 2 * A - 1$,

and c_k is defined as

$$\begin{aligned} c_k &= \frac{2}{n+1} \sum_{i=0}^{n} \log\left(I - x_i\right) T_k\left(x_i\right), \\ x_i &= \cos\left(\frac{\left(i + \frac{1}{2}\right)\pi}{n+1}\right). \end{aligned} \tag{7.20}$$

The Chebyshev approximation is appealing as it gives the best m-degree polynomial approximation of $\log\left(I - x\right)$ under the L_∞-norm. The error induced by general Chebyshev polynomial approximations has also been thoroughly investigated [78].

Stochastic Lanczos Quadrature: This approach [153] relies on stochastic trace estimation to approximate the trace using the identity presented in (7.1). If we consider the eigendecomposition of matrix A into $Q\Lambda Q^\top$, the quadratic form in the equation becomes

$$\begin{aligned} \mathbf{r}^{(i)\top} \log(A) \mathbf{r}^{(i)} &= \mathbf{r}^{(i)\top} Q \log\left(\Lambda\right) Q^\top \mathbf{r}^{(i)} \\ &= \sum_{k=1}^{n} \log\left(\lambda_k\right) \mu_k^2 \end{aligned} ,$$

where μ_k denotes the individual components of $Q^\top \mathbf{r}^{(i)}$. By transforming this term into a Riemann-Stieltjes integral $\int_a^b \log(t) \mathrm{d}\mu(t)$, where $\mu(t)$ is a piecewise constant function [153], we can approximate it as

$$\int_a^b \log(t) \mathrm{d}\mu(t) \approx \sum_{j=0}^{m} \omega_j \log\left(\theta_j\right),$$

where m is the degree of the approximation, while the sets of ω and θ are the parameters to be inferred using Gauss quadrature. It turns out that these parameters may be computed analytically using the eigendecomposition of the low-rank tridiagonal transformation of A obtained using the Lanczos algorithm [121]. Denoting the resulting eigenvalues and eigenvectors by θ and y respectively, the quadratic

form may finally be evaluated as,

$$\mathbf{r}^{(i)\top} \log(A)\, \mathbf{r}^{(i)} \approx \sum_{j=0}^{m} \tau_j^2 \log(\theta_j), \tag{7.21}$$

with $\tau_j = \left[e_1^T y_j\right]$.

7.4.1 Synthetically Constructed Matrices

Previous work on estimating log-determinants have implied that the performance of any given method is closely tied to the shape of the eigenspectrum for the matrix under review. As such, we set up an experiment for assessing the performance of each technique when applied to synthetically constructed matrices whose eigenvalues decay at different rates. Given that the computational complexity of each method is dominated by the number of matrix-vector products (MVPs) incurred, we also illustrate the progression of each technique for an increased allowance of MVPs. All matrices are constructed using a Gaussian kernel evaluated over 1000 input points.

As illustrated in Figure 7.2, the estimates returned by our approach are consistently on par with (and frequently superior to) those obtained using other methods. For matrices having slowly-decaying eigenvalues, standard Chebyshev and Taylor approximations fare quite poorly, whereas SLQ and our approach both yield comparable results. The results become more homogeneous across methods for faster-decaying eigenspectra, but our method is frequently among the top two performers. For our approach, it is also worth noting that truncating the GP using known bounds on the log-determinant indeed results in superior posterior estimates. This is particularly evident when the eigenvalues decay very rapidly. Somewhat surprisingly, the performance does not seem to be greatly affected by the number of budgeted MVPs.

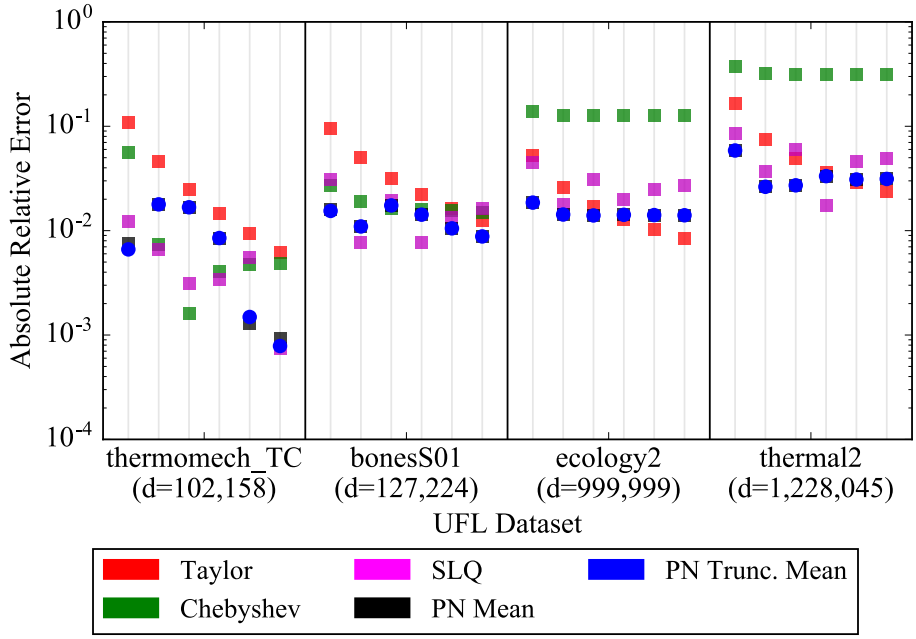


Figure 7.3: Methods compared on a variety on UFL Sparse Datasets. Each dataset was ran the matrix approximately raised to the power of 5, 10, 15, 20, 25 and 30 (left to right) using stochastic trace estimation. Absolute relative error is defined as the absolute difference between the truth and predicted, divided by the truth.

7.4.2 UFL Sparse Datasets

Although we have so far limited our discussion to covariance matrices, our proposed method is amenable to any positive semi-definite matrix. To this end, we extend the previous experimental set-up to a selection of real, sparse matrices obtained from the SuiteSparse Matrix Collection [42]. Following [153], we list the true values of the log-determinant reported in [24], and compare all other approaches to this baseline.

The results for this experiment are shown in Figure 7.3. Once again, the estimates obtained using our probabilistic approach achieve comparable accuracy to the competing techniques, and several improvements are noted for larger allowances of MVPs. As expected, the SLQ approach generally performs better than Taylor and Chebyshev approximations, especially for smaller computational bud-

gets. Even so, our proposed technique consistently appears to have an edge across all datasets.

7.4.3 Uncertainty Quantification

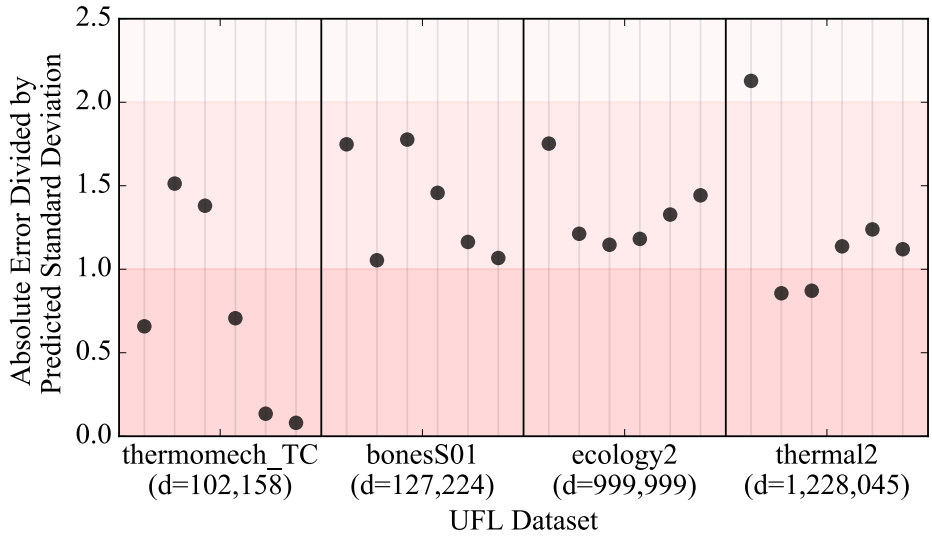


Figure 7.4: Quality of uncertainty estimates on UFL datasets, measured as the ratio of the absolute error to the output variance of the Gaussian process. As before, results are shown for increasing computational budgets (MVPs). The true value lay outside 2 standard deviations in only one of 24 trials.

One of the notable features of our proposal is the ability to quantify the uncertainty of the predicted log-determinant, which can be interpreted as an indicator of the quality of the approximation. Given that none of the other techniques offers such insights to compare against, we assess the quality of the model's uncertainty estimates by measuring the ratio of the absolute error to the predicted standard deviation (uncertainty). For the latter to be meaningful, the error should ideally lie within only a few multiples of the standard deviation.

In Figure 7.4, we report this metric for our approach when using the histogram

kernel. We carry out this evaluation over the matrices introduced in the previous experiment, once again showing how the performance varies for different MVPallowances. In all cases, the absolute error of the predicted log-determinant is consistently bounded by at most twice the predicted standard deviation, which is very sensible for such a probabilistic model.

7.4.4 Motivating Example

Determinantal point processes [106] are stochastic point processes defined over subsets of data such that an established degree of repulsion is maintained. A DPP, $\mathcal{P}$, over a discrete space $y \in \{1,\ldots,n\}$ is a probability measure over all subsets of y such that

$$\mathcal{P}(A \in y) = \mathrm{Det}(K_A),$$

where K is a positive definite matrix having all eigenvalues less than or equal to 1. A popular method for modelling data via K is the *L-ensemble* approach [22], which transforms kernel matrices, L, into an appropriate K,

$$K = (L + I)^{-1}L.$$

The goal of inference is to correctly parameterize L given observed subsets of y, such that the probability of unseen subsets can be accurately inferred in the future.

Given that the log-likelihood term of a DPP requires the log-determinant of L, naïve computations of this term are intractable for large sample sizes. In this experiment, we demonstrate how our proposed approach can be employed to the purpose of parameter optimization for large-scale DPPs. In particular, we sample points from a DPP defined on a lattice over $[-1,1]^5$, with one million points at uniform intervals. A Gaussian kernel with lengthscale parameter l is placed over

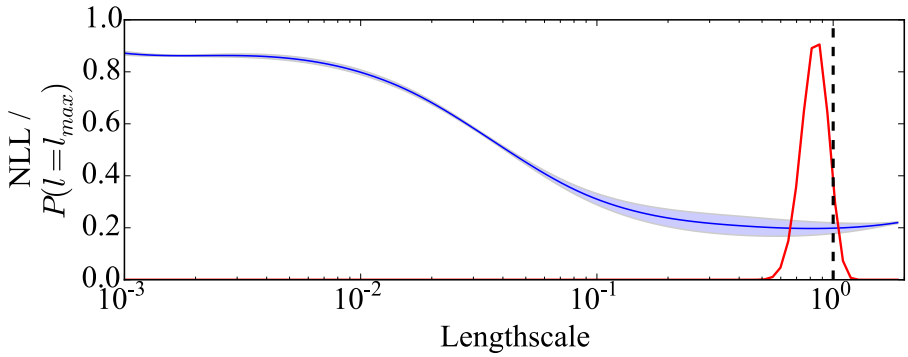


Figure 7.5: The rescaled Negative log likelihood (NLL) of DPP with varying length scale (blue) and the probability of maximum likelihood (red). Cubic interpolation was used between inferred likelihood observations. Ten samples, z, were taken to polynomial order 30.

these points, creating the true L. Subsets of the lattice points can be drawn by taking advantage of the tensor structure of L, and we draw five sets of 12,500 samples each. For a given selection of lengthscale options, the goal of this experiment is to confirm that the DPP likelihood of the obtained samples is indeed maximized when L is parameterized by the true lengthscale, l. As shown in Figure 7.5, the computed uncertainty allows us to derive a distribution over the true lengthscale which, despite using few matrix-vector multiplications, is very close to the optimal.

7.5 Conclusion

This chapter introduces a truly novel view of the eigen-spectrum as a density over the distribution of eigen-values. In a departure from conventional approaches for estimating the log-determinant of a matrix, we propose a novel probabilistic framework which provides a Bayesian perspective on the literature of matrix theory and

stochastic trace estimation while able to integrate upper and lower bounds on the log-determinant found in the linear algebraic literature. In particular, our approach enables the log-determinant to be inferred from noisy observations of $\mathrm{Tr}\left(A^k\right)$ obtained from stochastic trace estimation. By modelling these observations using a GP, a posterior estimate for the log-determinant may then be computed using Bayesian Quadrature. Our experiments confirm that the results obtained using this model are highly comparable to competing methods, with the additional benefit of measuring uncertainty. This adds credence to the suitability of our approach within a practical setting.

We forecast that the foundations laid out in this work can be extended in various directions, such as exploring more kernels on the raw moments which permit tractable Bayesian Quadrature. The uncertainty quantified in this work is also a step closer towards fully characterizing the uncertainty associated with approximating large-scale kernel-based models.

Chapter 8

Entropic Eigen-Spectrum Inference

As discussed in the previous chapter, the scalable calculation of matrix determinants has been a bottleneck to the widespread application of many machine learning methods such as determinantal point processes, Gaussian processes, generalised Markov random fields, graph models and many others. In this chapter, we estimate log determinants under the framework of maximum entropy, given information in the form of moment constraints from stochastic trace estimation. The estimates demonstrate a significant improvement on the state-of-the-art alternative methods, as shown on a wide variety of UFL sparse matrices. By taking the example of a general Markov random field, we also demonstrate how this approach can significantly accelerate inference in large-scale learning methods involving the log determinant.

In this chapter, we present an alternative probabilistic approximation of log determinants rooted in information theory, as opposed to Bayesian inference seen in the previous chapter, which exploits the relationship between stochastic trace estimation and the moments of a matrix's eigenspectrum. These estimates are used as moment constraints on the probability distribution of eigenvalues. This is achieved by maximising the entropy of the probability density $p(\lambda)$ given our moment constraints. In our inference scheme, we circumvent the issue inherent to the Gaus-

sian process approach [57], whereby positive probability mass may occur in the region of negative densities. In contrast, our proposed entropic approach implicitly encodes the constraint that densities are necessarily positive. Given equivalent moment information, we achieve competitive results on matrices obtained from the SuiteSparse Matrix Collection [42] which consistently outperform competing approximations to the log-determinant [57, 25].

The most significant contributions of this chapter are listed below.

1. We develop a novel approximation to the log-determinant of a matrix which relies on the principle of maximum entropy enhanced with moment constraints derived from stochastic trace estimation.

2. We directly compare the performance of our entropic approach to other state-of-the-art approximations to the log-determinant. This evaluation covers real sparse matrices obtained from the SuiteSparse Matrix Collection [42].

3. Finally, to showcase how the proposed approach may be applied in a practical scenario, we incorporate our approximation within the computation of the log-likelihood term of a Gaussian Markov random field, where we obtain a significant increase in speed.

8.1 Raw Moments of the Eigenspectrum

The relation between the raw moments of the eigenvalue distribution and the trace of matrix powers allows us to exploit stochastic trace estimation for estimating the log determinant. Raw moments are defined as the mean of the random variables raised to integer powers. Given that the function of a matrix is implicitly applied to

its eigenvalues, in the case of matrix powers, this corresponds to raising the eigenvalues to a given power. For example, the k^{th} raw moment of the distribution over the eigenvalues (a mixture of Dirac delta functions) is $\sum_{i=1}^{m} \lambda^k p(\lambda)$, where $p(\lambda)$ is the distribution of eigenvalues. The first few raw moments of the eigenvalues are trivial to compute. Denoting the kth raw moment as $\mathbb{E}[\lambda^k]$, we have that $\mathbb{E}[\lambda^0] = 1$, $\mathbb{E}[\lambda^1] = \frac{1}{n}\text{Tr}(A)$ and $\mathbb{E}[\lambda^2] = \frac{1}{n}\sum_{i,j} A_{i,j}^2$. More generally, the k^{th} raw moment can be formulated as $\mathbb{E}[\lambda^k] = \frac{1}{n}\text{Tr}(A^k)$, which can be estimated using stochastic trace estimation. These identities can be easily derived using the definitions and well-known identities of the trace term and Frobenius norm.

8.2 Approximating the Log Determinant

In view of the relation presented in the previous subsection, we can reformulate the log determinant of a matrix in terms of its eigenvalues using the following derivation:

$$\log\big(\text{Det}(A)\big) = \sum_{i=1}^{n} \log(\lambda_i) n\mathbb{E}\left[\log(\lambda)\right] \approx n \int p(\lambda)\log(\lambda)\text{d}\lambda \tag{8.1}$$

where the approximation is introduced due to our estimation of $p(\lambda)$, the probability distribution of eigenvalues. If we knew the true distribution of $p(\lambda)$ it would hold with equality. This general concept is the same as that utilised in the previous chapter.

Given that we can obtain information about the moments of $p(\lambda)$ through stochastic trace estimation, we can solve this integral by employing the principle of maximum entropy, while treating the estimated moments as constraints. While not explored in this work, it is worth noting that in the event of moment information combined with samples of eigenvalues, we would use the method of maximum relative entropy with data constraints, which is, in turn, a generalisation of Bayes'

rule [36]. This can be applied, for example, in the quantum linear algebraic setting [115].

8.3 Estimating the Log Determinant using Maximum Entropy

The maximum entropy method (MaxEnt) [125] is a procedure for generating the most conservatively uncertain estimate of a probability distribution possible with the given information, which is particularly valued for being maximally non-committal with regard to missing information [87]. In particular, to determine a probability density $p(\mathbf{x})$, this corresponds to maximising the functional

$$S = -\int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} - \sum_i \alpha_i \left[\int p(\mathbf{x}) f_i(\mathbf{x}) d\mathbf{x} - \mu_i \right] \tag{8.2}$$

with respect to $p(\mathbf{x})$, where $\mathbb{E}[f_i(\mathbf{x})] = \mu_i$ are given constraints on the probability density. In our case the mu_i are the ith stochastic trace estimates of $(1/n)M^i$, corresponding to the eigenvalue raw moment estimates, where M is the nxn matrix of interest. The first term in the above equation is referred to as the Boltzmann-Shannon-Gibbs (BSG) entropy, which has been applied in multiple fields, ranging from condensed matter physics [67] to finance [113, 31]. Along with its path equivalent, maximum calibre [70], it has been successfully used to derive statistical mechanics [73], non-relativistic quantum mechanics, Newton's laws and Bayes' rule [70, 36]. Under the axioms of consistency, uniqueness, invariance under coordinate transformations, sub-set and system independence, it can be proved that for constraints in the form of expected values, drawing self-consistent inferences requires maximising the entropy [143, 125]. Crucial for our investigation are the functional forms $f_i(\mathbf{x})$ of constraints for which the method of maximum entropy is appropriate. The axioms of Johnson and Shore [143] assert that the entropy must have

a unique maximum and that the BSG entropy is convex. The entropy hence has a unique maximum provided that the constraints are convex. This is satisfied for any polynomial in $\mathbf{x}$ and hence, maximising the entropy given moment constraints constitutes a self-consistent inference scheme [125].

8.3.1 Implementation

Our implementation of the system follows straight from stochastic trace estimation to estimate the raw moments of the eigenvalues, maximum entropy distribution given these moments and, finally, determining the log of the geometric mean of this distribution. The log geometric mean is an estimate of the log-determinant divided by the dimensionality of $A \in \mathbb{R}^{n \times n}$. We explicitly step through the subtleties of the implementation in order to guide the reader through the full procedure.

By taking the partial derivatives of S from Equation (8.2), it is possible to show that the maximum entropy distribution given moment information is of the form

$$p(\lambda) = \exp(-1 + \sum_i \alpha_i \mu_i).$$

The goal is to find the set of α_i which match the raw moments of $p(\lambda)$ to the observed moments $\{\mu_i\}$. While this *may* be performed symbolically, this becomes intractable for a larger number of moments, and our experience with current symbolic libraries [110, 165] is that they are not extendable beyond more than 3 moments. Instead, we turn our attention to numerical optimisation. Early approaches to optimising maximum entropy coefficients worked well for a small number of coefficients but became highly unstable as the number of observed moments grew [111]. However, building on these concepts more stable approaches emerged [14]. Algorithm 2 outlines a stable approach to this optimisation under the conditions that λ_i is strictly positive and the moments lie between zero and one. We can sat-

isfy these conditions by normalising our positive definite matrix by the maximum of the Gershgorin intervals [65].

Algorithm 2 Optimising the Coefficients of the MaxEnt Distribution

Input: Moments $\{\mu_i\}$, Tolerance ϵ
Output: Coefficients $\{\alpha_i\}$
1: $\alpha_i \sim \mathcal{N}(0,1)$
2: $i \leftarrow 0$
3: $p(\lambda) \leftarrow \exp(-1 - \sum_k \alpha_k \lambda^k)$
4: **while** error $< \epsilon$ **do**
5: $\quad \delta \leftarrow \log\left(\frac{\mu_i}{\int \lambda^i p(\lambda) d\lambda}\right)$
6: $\quad \alpha_i \leftarrow \alpha_i + \delta$
7: $\quad p(\lambda) \leftarrow p(\lambda|\alpha)$
8: $\quad$ error $\leftarrow \max|\int \lambda^i p(\lambda) d\lambda - \mu_i|$
9: $\quad i \leftarrow \text{mod}(i+1, \text{length}(\mu))$

Given Algorithm 2, the pipeline of our approach can be pieced together. First, the raw moments of the eigenvalues are estimated using stochastic trace estimation. These moments are then passed to the maximum entropy optimisation algorithm to produce an estimate of the distribution of eigenvalues, $p(\lambda)$. Finally, $p(\lambda)$ is used to estimate the log geometric mean of the distribution, $\int \log(\lambda) p(\lambda) d\lambda$. This term is multiplied by the dimensionality of the matrix and if the matrix is normalised, the log of this normalisation term is added again. These steps are laid out more concisely in Algorithm 3.

Algorithm 3 Entropic Trace Estimation for Log Determinants

Input: PD Symmetric Matrix A, Order of stochastic trace estimation k, Tolerance ϵ
Output: Log Determinant Approximation $\log|A|$
1: $B = A/\|A\|_2$
2: μ (moments)$\leftarrow$ StochasticTraceEstimation(B,k)
3: α (coefficients) $\leftarrow$ MaxEntOpt(μ,ϵ)
4: $p(\lambda) \leftarrow p(\lambda|\alpha)$
5: $\log|A| \leftarrow n\int \log(\lambda) p(\lambda) d\lambda + n\log(\|A\|_2)$

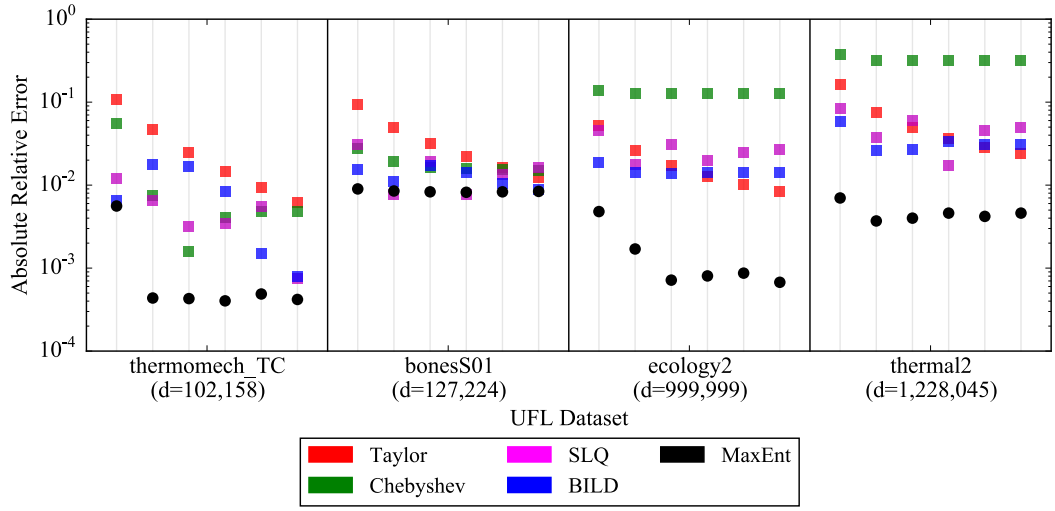


Figure 8.1: Comparison of competing approaches over four UFL datasets. Results are also shown increasing computational budgets, i.e. 5, 10, 15, 20, 25 and 30 moments respectively. Our method obtains substantially lower error rates across 3 out of 4 datasets, and still performs very well on 'bonesS01'.

8.4 Experiments

So far, we have supplemented the theoretic foundations of our proposal by devising experiments on synthetically constructed matrices. In this section, we extend our evaluation to include real matrices obtained from a variety of problem domains and demonstrate how the results obtained using our approach consistently outperform competing state-of-the-art approximations. Moreover, in order to demonstrate the applicability of our method within a practical domain, we highlight the benefits of replacing the exact computation of the log-determinant term appearing in the log-likelihood of a Gaussian Markov random field with our maximum entropy approximation.

8.4.1 UFL Datasets

While the ultimate goal of this work is to accelerate inference of large-scale machine learning algorithms burdened by the computation of the log-determinant,

Table 8.1: Comparison of competing approximations to the log-determinant over additional sparse UFL datasets. The technique yielding the lowest relative error is highlighted in bold, and our approach is consistently superior to the alternatives. Approximations are computed using 10 moments estimated with 30 probing vectors.

Dataset	Dimension	Taylor	Chebyshev	SLQ	BILD	MaxEnt
shallow_water1	81,920	**0.0023**	0.7255	0.0058	0.0163	0.0030
shallow_water2	81,920	0.5853	0.9846	0.9385	1.1054	**0.0051**
apache1	80,800	0.4335	0.0196	0.4200	0.1117	**0.0057**
finan512	74,752	0.1806	0.1158	0.0142	**0.0005**	0.0171
obstclae	40,000	0.0503	0.5269	0.0423	0.0733	**0.0026**
jnlbrng1	40,000	0.1084	0.2079	0.0465	0.0805	**0.0158**

this is a general approach which can be applied to a wide variety of application domains. The SuiteSparse Matrix Collection [42] (commonly referred to as the set of UFL datasets) is a collection of sparse matrices obtained from various real problem domains. In this section, we shall consider a selection of these matrices as 'matrices in the wild' for comparing our proposed algorithm against established approaches. In this experiment we compare against Taylor [7] and Chebyshev [79] approximations, stochastic Lanczos quadrature (SLQ) [154] and Bayesian inference of log determinants (BILD) [57]. In Figure 8.1, we report the absolute relative error of the approximated log determinant for each of the competing approaches over four different UFL datasets. Following [57], we assess the performance of each method for an increasing computational budget, in terms of matrix-vector multiplications, which in this case corresponds to the number of moments considered. It can be immediately observed that our entropic approach vastly outperforms the competing techniques across all datasets, and for any given computational budget. The overall accuracy also appears to consistently improve when more moments are considered.

Complementing the previous experiment, Table 8.1 provides a further comparison on a range of other sample matrices which are large, yet whose determinants can be computed by standard machines in a reasonable time (by virtue of being sparse). For this experiment, we consider 10 estimated moments using 30 probing vectors, and their results are reported for the aforementioned techniques. The results presented in Table 8.1 are the relative error of the log determinants *after* they have been normalised using Gershgorin intervals [65]. We note, however, that the methods improve at different rates as more raw moments are taken.

8.4.2 Computation of GMRF Likelihoods

Gaussian Markov random fields (GMRFs) [136] specify spatial dependence between nodes of a graph with Markov properties, where each node denotes a random variable belonging to a multivariate joint Gaussian distribution defined over the graph. These models appear in a wide variety of applications, ranging from interpolation of spatiotemporal data to computer vision and information retrieval. While we refer the reader to [136] for a more comprehensive review of GMRFs, we highlight the fact that the model relies on a positive-definite precision matrix Q_θ parameterised by θ, which defines the relationship between connected nodes; given that not all nodes in the graph are connected, we can generally expect this matrix to be sparse. Nonetheless, parameter optimisation of a GMRF requires maximising the following equation:

$$\log p(\mathbf{x} \mid \theta) = \frac{1}{2}\log\big(\mathrm{Det}(Q_\theta)\big) - \frac{1}{2}\mathbf{x}^\top Q_\theta \mathbf{x} - \frac{n}{2}\log(2\pi)$$

where computing the log determinant poses a computational bottleneck, even where Q_θ is sparse. This arises because it is possible for the Cholesky decomposition of a sparse matrix with zeros outside a band of size k to be nonetheless

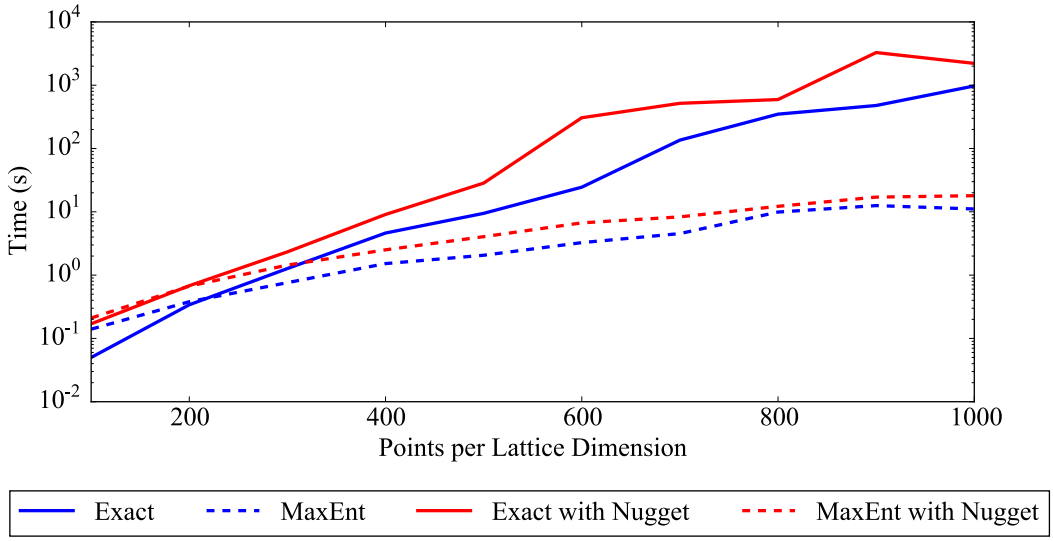


Figure 8.2: Time in seconds for computing the log likelihood of a GMRF via Cholesky decomposition or using our proposed MaxEnt approach for estimating the log determinant term. Results are shown for GMRFs constructed on square lattices with increasing dimensionality, with and without a nugget term.

dense *within* that bound. Thus, the Cholesky decomposition is still expensive to compute.

Following the experimental set-up and code provided in [74], in this experiment we evaluate how incorporating our approximation into the log-likelihood term of a GMRF improves scalability when dealing with large matrices, while still maintaining precision. In particular, we construct lattices of increasing dimensionality and in each case measure the time taken to compute the log-likelihood term using both approaches. The precision kernel is parameterised by κ and τ [104], and is explicitly linked to the spectral density of the Matérn covariance function for a given smoothness parameter. We repeat this evaluation for the case where a nugget term, which denotes the variance of the non-spatial error, in included in the constructed GMRF model. Note that for the maximum entropy approach we employ 30 sample vectors in the stochastic trace estimation procedure, and consider 10 moments. As illustrated in Figure 8.2, the computation of the log likelihood is

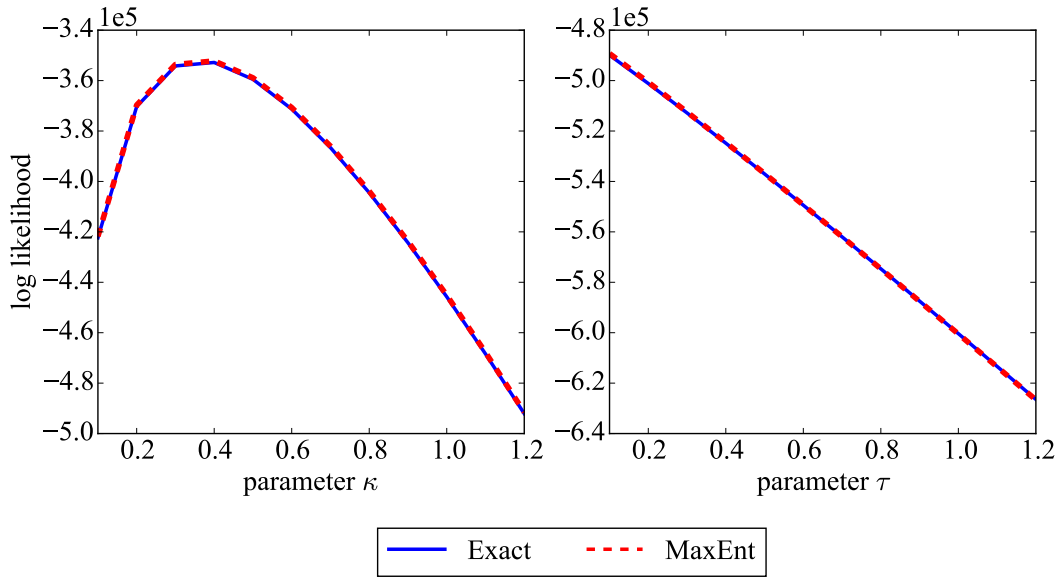


Figure 8.3: The above plots indicate the difference of log likelihood between the exact computation of the likelihood and the maximum entropy approach for a range of hyperparameters of the model. We note that the extrema of both exact and approximate inference align and it is difficult to distinguish the two lines.

orders of magnitude faster when computing the log-determinant using our proposed maximum entropy approach. In line with our expectations, this speed-up is particularly significant for larger matrices. Similar improvements are observed when a nugget term is included. Note that we set $\kappa = 0.1$ and $\tau = 1$ for this experiment.

Needless to say, improvements in computation time mean little if the quality of inference degrades. Figure 8.3 illustrates the comparable quality of the log likelihood for various settings of κ and τ, and the results confirm that our method enables faster inference without compromising on performance.

8.5 Conclusions

Inspired by the probabilistic interpretation introduced in the previous chapter and [57], in this chapter we have developed a novel approximation to the log-determinant

which is rooted in information theory. While lacking the uncertainty quantification inherent to the aforementioned technique, this formulation is appealing because it uses a comparatively less informative prior on the distribution of eigenvalues, and we have also demonstrated that the method is theoretically expected to yield superior approximations for matrices of very large dimensionality. This is especially significant given that the primary scope for undertaking this work was to accelerate the log-determinant computation in large-scale inference problems. As illustrated in the experimental section, the proposed approach consistently outperforms all other state-of-the-art approximations by a sizeable margin. The work from this chapter lead to the 2017 ECML publication as outlined in chapter 1.

8.6 Sparse Linear Algebra for Gaussian Processes

I have now introduced a range of techniques which leverage sparse linear algebra as a speed up to the core operations of Gaussian process inference. In this short chapter to conclude the section, I have brought these together into a full Gaussian process pipeline. For inverse quadratic forms, I have used conjugate gradient descent to solve the system of equations and for log determinants, the MaxEnt approach described in this chapter was used.

As a benchmark, I have compared the results to the FITC as described in Chapter 5, as such sparse approximations are often the *go-to* for practitioners who wish to speed up their inference. However, this flavour of approximation endeavours to replace the full covariance matrix with a near-optimal low-rank approximation. This works well when the system has low degrees of freedom but, as we shall see in this section, performs poorly when the eigenvalue decay rate is gentle.

The dataset chosen for this demonstration is inspired by transfer learning. I first have trained a simple convolutional neural network on the MNIST dataset in Keras. The neural network achieves 98.86% accuracy on a held-out testing set. The goal of this regression task is to mimic the dense layers of the network using a Gaussian process with the output fitting to the output logits of the network (ie the output of the neural network before softmax is applied). As the input of the Gaussian process is the output of the convolutional layers, the input dimension is 4608 which is significantly larger than the original 28x28 pixels of the dataset. As we are endeavouring to mimic the behaviour of a neural network the kernel is chosen as a dot product kernel with some white noise added. The logits were normalised to remove the mean and have the variance set to one. To keep the runtime of the experiments reasonably fast the dataset used was of 3,000 MNIST

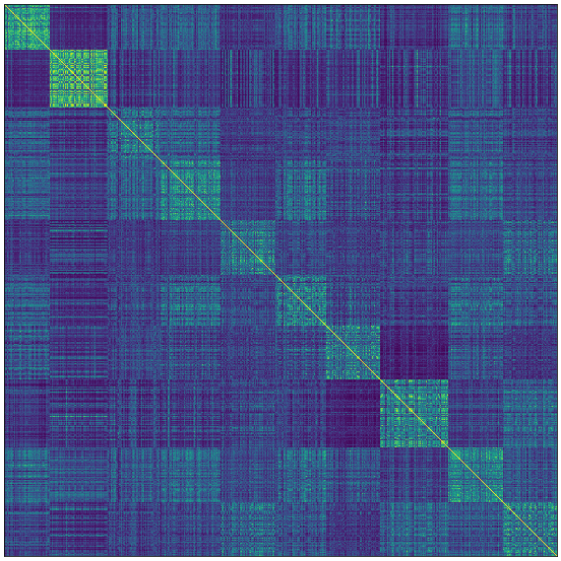


Figure 8.4: The image on the left is a visualisation of the dot product kernel when all hyper parameters are set to ones. For visualisation purposed dark blue represents small numbers and yellow represents large values. The data has been ordered based on the true label of the data allowing us to see the dominant block-wise diagonal nature of the matrix. On the right is a visualisation of the eigenvalues of the same matrix.

characters.

While this could be seen as a multi-output Gaussian process, considering all classes simultaneously, I instead kept the demonstration simple by considering each logit as a separate learning task. In order to present meaningful results, the predicted output is then passed through a logistic function and thresholded. Figure 8.6 shows a principal 1000x1000 matrix from the kernel matric and its Eigen spectrum. As can be seen, the eigenvalue decay slowly.

In the proposed approximation the kernel matrix itself is not sparse, however, I chose an arbitrary threshold of 0.05 and clipped all of the input data below the threshold to zero. The results in an input data matrix which is 97.2% sparse. As both the conjugate gradient solver and the log determinant rely on dot product and as $K = X\Theta^2X^T$, with Θ being a diagonal matrix of hyper-parameters to be

learned during training, multiplication of a vector with K requires no dense matrix multiplication.

	Proposed Approach		FITC	
	True Positives	Accuracy	True Positives	Accuracy
0	**85.71%**	**98.48%**	75.00%	98.23%
1	**97.36%**	**99.49%**	86.84%	98.73%
2	**75.67%**	**96.71%**	48.64%	95.20%
3	**81.63%**	**97.22%**	61.22%	95.20%
4	**86.48%**	**97.97%**	78.38%	97.97%
5	**85.00%**	**97.72%**	60.00%	95.95%
6	**75.00%**	**97.47%**	66.67%	96.97%
7	**95.65%**	**98.48%**	67.39%	96.21%
8	**93.54%**	**97.97%**	67.74%	97.22%
9	**86.53%**	**96.72%**	67.31%	95.20%

As can be seen from the above table the approach proposed of using sparse linear algebra appears to consistently outperform the low-rank approach. This result, of course, does not necessarily translate to all problems about this problem was chosen specifically to demonstrate the use of sparse linear algebra knowing that there was a large number of degrees of freedom in the kernel matrix.

Part IV

Towards the Future of Machine Learning

Chapter 9

Quantum Computing for Machine Learning

While the machine learning community has been making huge progress to scale algorithms to large datasets, the dawn of quantum computing technology may be approaching and offers potentially exponential speed ups to many of the bottlenecks we face. This chapter introduces some such quantum algorithms at a high-level and shows how they may be applied in the field of machine learning.

Quantum computers naturally store the state of a system as a vector over a complex field. Computation is performed by applying sets of unitary operations to the system and computational complexity is calculated based on how easily these unitary operations can be constructed from a base of primitives. In essence, quantum computers apply a set of gate operations on qubits the way a classical computer applies an alternative set of gates on classical bits. Fortunately, as machine learning researchers we rarely worry too much about the silicon in the CPU or GPU of our computer, but rather determine algorithmic advances purely using primitives such as the cost of matrix arithmetic. When it comes quantum machine learning this needs to be no different. We will endeavour to understand some of the basic results in quantum computing and apply them to the problems we face

in our field.

For ease of access, some of the quantum speed ups for common linear algebraic tasks are:

Fast loading of data into state vector [68]:	$O(1)$
Fast Fourier transform [115]:	$O((\log N)2)$
Sparse Matrix operations:	$O(poly(\log(N)))$
Trace/Det estimation [172]:	$O(poly(\log(N)))$
Inner Product:	$O(\log(N))$
Linear Solver [81]:	$O(poly(\log(N)))$
Sampling Eigenvalues [97]:	$O(poly(\log(N)))$

9.1 The Qubit

As mentioned in the chapter introduction, quantum computing is a generalisation from classical computing which we are a costumed to whereby bits are replaced with their complex counterpart, the qubit. While this presents new challenges for experimental physicists to build at scale in practice, it offers a wide variety of opportunities from an algorithmic design perspective. Quantum states are represented as unit vectors in complex Hilbert spaces and are combined via tensor products. The general procedure to run a quantum algorithm is to first set up the initial state, to then apply a number of unitary operations to the system and then to measure the result. Results are inherently probabilistic and we measure the probability of the resulting system ψ being in state i as $\psi^T i$, or $< \psi | i >$ in braket notation (as the physicists more commonly use). Quantum states change over time such that $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$, where H is the *Hamiltonian* of the system. Some of the common gates applied to quantum systems, or more importantly the primitives to

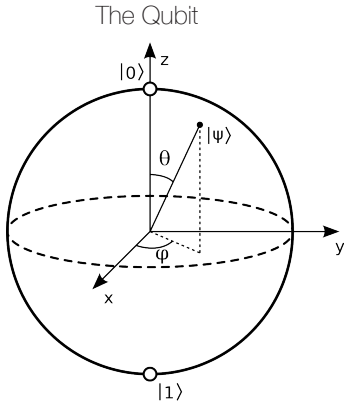


Figure 9.1: Above is a visual diagram of a single qubit taken from Wikipedia.

most of the quantum algorithms discussed are:

Hadamard	Phase	$\pi/8$	CNOT	CZ
$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{\pi}{4}i} \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
				

The above primitives can be applied to large systems by taking their tensor product with the identity over other subsystems. As can be seen, some gates above apply to single qubits while others act on entangled pairs. Similarly to how classical NAND gates are universal for classical computers, many combinations of quantum gates are also sufficient for quantum universality [115]. For example, if we have single qubit gates along with an entangling 2-qubit gate we have a universal logical set.

9.2 Quantum Random Access Memory (QRAM)

Quantum Random Access Memory (QRAM) [68] is the coined term for the quantum equivalent to RAM. Classically we are used to memory by a pointer to retrieve

values stored and the quantum analogy is no different allowing the system to read memory from quantum superpositions,

$$\sum_{i,j} \alpha_{ij} |i\rangle |j\rangle \xrightarrow{\text{QRAM}} \sum_{i,j} \alpha_{ij} |i\rangle |j + m_i\rangle$$

where m_i is the ith entry stored in memory and $|i\rangle$ is the ith basis vector. QRAM provides a way to probabilistically produce $|\mathbf{x}\rangle$ for any $\mathbf{x}$ stored in memory. The steps to get a prepared vector $|\mathbf{x}\rangle$ for any $\mathbf{x}$ from memory, we start the state $D^{-\frac{1}{2}} \sum_i |i\rangle |0\rangle$ as the query state for the QRAM to obtain $D^{-\frac{1}{2}} \sum_i |i\rangle |x_i\rangle$, where $D^{-\frac{1}{2}}$ is a normalising factor. Essentially we are rotating the auxiliary vector $|0\rangle$ into the correct position. A key term mentioned here is that we are *probabilistically* producing $|\mathbf{x}\rangle$. While QRAM is incredibly useful, there was originally some debate over issues regarding a post-selection step required in practice. However, we have resolved these issues by shifting the quantization boundaries of the data by a half precision [170].

9.3 Phase Estimation

In the previous chapters, it was identified that linear algebra operation is the backbone of much of machine learning and regularly act as a bottleneck. Phase estimation [97] is a quantum algorithm to probabilistically estimate the eigenvalues of a matrix via a combination of Hadamard gates, controlled unitary estimations and inverse quantum Fourier transforms. From an algorithm design perspective, given a vector x and state ψ phase estimation allows us to sample the eigenvalues of ψ in proportion to the projection of x on the corresponding eigenvector. For example, imagine we wished to sample the eigenvalues of ψ uniformly at random we would apply phase estimation with a unit vector x chosen uniformly at random from the

hypersphere or equally a column of the identity matrix chosen at random. This can be performed in $O(poly(\log(N)))$ for all sparse and low-rank Hermitian matrices.

One of the key algorithms in quantum machine learning is the HHL algorithm [81] for solving linear systems of equations. It extends phase estimation via the use of an ancilla qubit and performs a controlled rotation which inverts the superposition of eigenvalues. Now, when the vector x is applied we observe an inverted eigenvalue selected in proportion to the projection of x or the corresponding eigenvector. This acts as a Monte Carlo sample of $\psi^{-1}x$ and will achieve a fixed precision outcome given a constant number of samples. For sparse and low rank Hermitian matrices with condition number κ, linear systems of equations of the form $A^{-1}b$ can be computed in $O(poly(\log(N))\kappa^2)$. Further, the authors of the HHL algorithm show that the result holds for any function of the eigenvalues.

9.4 Quantum Gaussian Process Training and Inference

Gaussian process regression requires the efficient calculation of three core equations, namely the mean estimate, variance estimate and the negative log marginal likelihood (NLML) used during training. Assuming a zero-mean prior we must solve,

$$\begin{aligned} E[f(x^*)] &= K(x^*,x)K(x,x)^{-1}y \\ V[f(x^*)] &= K(x^*,x^*) - K(x^*,x)K(x,x)^{-1}K(x,x^*) \\ \text{NLML} &= -\frac{1}{2}y^T K(x,x)^{-1}y - \frac{1}{2}\log|K(x,x)| + C \end{aligned} \tag{9.1}$$

In [171, 173] we break these down into the core linear algorithmic operations a provide variants of popular quantum linear algebraic operations in order to compute each component. Working backwards, $\log|K(x,x)|$ can be rewritten as the

trace of the log matrix $\mathrm{Tr}(\log(K(x,x)))$. The log matrix is defined as the matrix with the same eigen decomposition but with eigenvalues equal to the logarithm of those in $K(x,x)$. Intuitively we can modify the concepts from quantum phase estimation, which efficiently allow us to sample eigenvalues of Hermitian matrices in $O(poly(\log(N)))$, such that we sample eigenvalues and then compute their log hence building up a simple Monte Carlo estimate of the trace.

The remaining term of the negative log marginal likelihood, $y^T K(x,x)^{-1} y$, can also be computed with a variant of the HHL algorithm. In this setting we do not endeavour to find $K(x,x)^{-1}y$ as the vanilla HHL algorithm would, but rather $K(x,x)^{-\frac{1}{2}}y$. The algorithm has a success probability equal to the square of the result. As such, by monitoring the success probability of the computation we efficiently find $(K(x,x)^{-\frac{1}{2}}y)^2 = y^T K(x,x)^{-1} y$. Combing these first two results we are able to calculate the NLML and hence train our Gaussian process.

Once trained we likely wish to predict new values. This involves computing $K(x^*,x)K(x,x)^{-1}K(x,x^*)$ and $K(x^*,x)K(x,x)^{-1}y$. While the latter term may be computed in a similar way to $y^T K(x,x)^{-1} y$, in [171] we propose an alternative variant which leverages a proposed inner product trick to solve inverse quadratic systems of the form $u^T A^{-1} v$. The two main contributions in [171] are how to extend the standard QLA algorithm to efficiently take advantage of quantum architectures. Firstly is the preparation of the vectors to be post multiplied by the kernel matrix inverse which takes advantage of QRAM for preparing a s-sparse vector. The second element introduced is to efficiently determine the inner product required to evaluate the inverse quadratic form rather than purely solving systems of linear equations. While there exists a method for this previously, it loses information about the sign of the result. This can be achieved via a modification of the

state preparation procedure. These two modifications are incorporated into HHL, producing an efficient Monte Carlo sampler of a Bernoulli variable with expectation proportional to the desired inverse quadratic form.

9.5 Application to Other Machine Learning Models

While the focus of this chapter has been on Gaussian processes, the algorithms set forth can be applied to a wide range of machine learning models. For example, a DPP could be shown to be efficient on a quantum computer by leveraging phase estimation to build up simple Monte Carlo estimates of the log-determinants required during training. Equally, GMRFs could use almost identical quantum systems to perform inference efficiently. If the GMRF modelled a lattice structure the tensor nature of the covariance matrix could be further leveraged. While deep learning often requires convolutional filters and max-pooling, potentially inefficient on quantum computers, the use of model distillation may allow for them to be recast into dense networks with which a quantum speedup could once again be applied.

9.6 Conclusion

In this chapter, we give a brief overview of the potential use of quantum computing to accelerate Gaussian processes. While the chapter was written as very much a high-level summary in order to not conflict with my collaborator Jansen Zhao's thesis, who was first author on each of these works, there are two novel contributions which are touched upon in this chapter. The first was the use of a variant of the HHL algorithm to accelerate the computational complexity of predictive estimates for Gaussian processes. Secondly, we have proposed a phase-estimation

like algorithm in order to calculate the log determinant of kernel matrix. If and when quantum computers scale over the coming years to hundreds and thousands of qubits, it would appear that quantum based machine learning may radically change the field due to the exponential speedups in computational performance.

Chapter 10

Conclusion

The goal of the research throughout this thesis has been to look at alternative and future approaches to increasing the applicability of Gaussian processes and kernel methods through greater scalability and generalisation. It has been shown not only the importance of kernel methods in their own right but also the benefit of viewing other machine learning approaches such as decision trees and neural networks under the guise of kernel methods. As such, general claims can be made such as frameworks for fair regression.

In chapter 4, we introduce the first of the novel contributions of this thesis. The chapter contains three important contributions. Firstly, an open problem outlined in [46] has been addressed as we define a formal definition for group fairness in expectation for regression problems. I then clearly outline an approach which constrains a regressor such that the expectation of the of regressor with respect to two or more empirical distributions is equal. While this is highly effective, I note that there is a natural cost to individual fairness in this process. Nevertheless, this approach has likely multiple applications outside of the fair machine learning literature. Finally, it is shown that this approach works for all kernel regression which minimizes the L2 loss of the data. Hence, it is applicable to decision trees, Gaussian processes and even some neural networks. For fast models such as decision trees,

I have also proven that it has no additional computational or memory complexity over the traditional approaches. This chapter has formed a paper in Entropy, 2019.

Turning our attention from how general kernel methods can be seen to the issue of scalability, chapter 5 presents an overview of scalable approaches for Gaussian processes. We note that one direction which has received little attention is the approach of stochastic trace estimation (STE). In chapter 6, STE was reexamined and a novel state of the art sampling approach was formulated. This method leveraged a linear algebraic construct called mutually unbiased bases (MUBs) which is commonly used in the field of quantum mechanics but has received little attention within the machine learning community. Leveraging these MUBs to sample probing vectors, I have shown that a novel state-of-the-art approach to stochastic trace estimation can be formed. This both theoretically and empirically outperforms the competing methods and has formed the basis of my paper at UAI, 2018.

Off the back of this work, an investigation into probabilistic approaches for calculating log determinants was conducted in collaboration with Maurizio Filippone and Kurt Cutajar of Eurocom, Sophie Antipolis, France. This led to an approach using Gaussian processes to model the distribution of eigenvalues of kernel matrices as outlined in chapter 7. In this chapter, Gaussian processes are used in modelling the distribution of eigenvalue magnitudes. This is a radically new way of approaching log-determinant calculations and has shown an initial empirical success. This work formed the body of the UAI, 2017 paper as outlined in chapter 1.

Later, we found that using maximum entropy (MaxEnt) methods gave us far better empirical performance. In chapter 8, a thorough investigation of MaxEnt for the use of log-determinant calculations was examined which in turn lead to the

ECML, 2017 paper as outlined in chapter 1. The core difference between this work and that of chapter 7 was to replace the approximate modelling of the density of eigenvalues using a Gaussian process via moment observations, to using the moments to model the density with a maximum entropy distribution.

An orthogonal direction of research was also investigated, namely examining the potential of quantum computers for the acceleration of Gaussian processes and kernel methods. This work was done in collaboration with Zhikuan (Jansen) Zhao and Joseph Fitzsimons at Singapore's University of Technology and Design (SUTD) and the National University of Singapore's Centre for Quantum Technologies (CQT). The first effort in this direction of this was to examine the performance acceleration by using a quantum approach on the quadratic form used during predictions. Secondly, we investigated the ability to accelerate the log determinant calculation required to train Gaussian processes efficiently. By combining these we have provided a practical set of algorithms which can be applied on quantum computers. This along with the rapid progression of quantum hardware gives us a strong hope that we may see these algorithms applied in practice in the coming years.

Many areas of these areas of research have yet much scope for investigation. I have started work in the direction of transfer learning and distillation based on the premise of Hilbert space similarity metrics. Yet another example whereby taking a unifying view of kernel methods may lead to an effective advance in the field. There is also much scope for further numerical approaches to be developed. One of the more exciting of these is in the direction of hierarchical matrices for numerical approximations which I do not believe has received sufficient attention to date. Finally, quantum machine learning is still in its infancy and while many are apply-

ing well-known quantum speedups to classical machine learning algorithms, there is likely a lot of scope to develop bespoke quantum algorithms which further take advantages of the nuanced details of specific machine learning approaches. I hope to continue my work in these areas over the coming years.

Bibliography

[1] Sivaram Ambikasaran and Eric Darve. The inverse fast multipole method. *arXiv preprint arXiv:1407.1572*, 2014.

[2] Mihai Anitescu, Jie Chen, and Lei Wang. A Matrix-free Approach for Solving the Parametric Gaussian Process Maximum Likelihood Problem. *SIAM J. Scientific Computing*, 34(1), 2012.

[3] Nicholas Francis Arcolano. *Approximation of Positive Semidefinite Matrices Using the Nystrom Method*. PhD thesis, HARVARD UNIVERSITY, 2011.

[4] Mikhail J Atallah, Frédéric Chyzak, and Philippe Dumas. A randomized algorithm for approximate string matching. *Algorithmica*, 29(3):468–486, 2001.

[5] Mikhail J Atallah, Elena Grigorescu, and Yi Wu. A lower-variance randomized algorithm for approximate string matching. *Information Processing Letters*, 113(18):690–692, 2013.

[6] Erlend Aune, Daniel P Simpson, and Jo Eidsvik. Parameter Estimation in High Dimensional Gaussian Distributions. *Statistics and Computing*, 24(2):247–263, 2014.

[7] Erlend Aune, Daniel P Simpson, and Jo Eidsvik. Parameter Estimation in High Dimensional Gaussian Distributions. *Statistics and Computing*, 24(2):247–263, 2014.

[8] Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, volume 10, pages 10–9, 2010.

[9] Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):8, 2011.

[10] Haim Avron and Sivan Toledo. Randomized Algorithms for Estimating the Trace of an Implicit Symmetric Positive Semi-definite Matrix. *J. ACM*, 58(2):8:1–8:34, 2011.

[11] Zhaojun Bai, Mark Fahey, Gene H Golub, M Menon, and E Richter. Computing partial eigenvalue sums in electronic structure calculations. Technical report, Citeseer, 1998.

[12] Zhaojun Bai and Gene H. Golub. Bounds for the Trace of the Inverse and the Determinant of Symmetric Positive Definite Matrices. *Annals of Numerical Mathematics*, 4:29–38, 1997.

[13] Maria-Florina Balcan, Travis Dick, Ritesh Noothigattu, and Ariel D Procaccia. Envy-free classification. *arXiv preprint arXiv:1809.08700*, 2018.

[14] K Bandyopadhyay, Arun K Bhattacharya, Parthapratim Biswas, and DA Drabold. Maximum entropy and the problem of moments: A stable algorithm. *Physical Review E*, 71(5):057701, 2005.

[15] Somshubhro Bandyopadhyay, P Oscar Boykin, Vwani Roychowdhury, and Farrokh Vatan. A new proof for the existence of mutually unbiased bases. *Algorithmica*, 34(4):512–528, 2002.

[16] Ronald Paul Barry and R Kelley Pace. Monte Carlo Estimates of the Log-Determinant of Large Sparse Matrices. *Linear Algebra and its applications*, 289(1):41–54, 1999.

[17] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24. ACM, 2008.

[18] Yahav Bechavod, Katrina Ligett, Aaron Roth, Bo Waggoner, and Zhiwei Steven Wu. Equal opportunity in online classification with partial feedback. *arXiv preprint arXiv:1902.02242*, 2019.

[19] 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.

[20] Steffen Börm and Jochen Garcke. Approximating gaussian processes with $\mathcal{H}^2$-matrices. In *Machine Learning: ECML 2007*, pages 42–53. Springer, 2007.

[21] Steffen Börm, Lars Grasedyck, and Wolfgang Hackbusch. Introduction to hierarchical matrices with applications. *Engineering Analysis with Boundary Elements*, 27(5):405–422, 2003.

[22] Alexei Borodin. Determinantal point processes. *arXiv preprint arXiv:0911.1153*, 2009.

[23] Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, and Anastasios Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *arXiv preprint arXiv:1503.00374*, 2015.

[24] Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, and Anastasios Zouzias. A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix. *CoRR*, abs/1503.00374, 2015.

[25] Christos Boutsidis, Petros Drineas, Prabhanjan Kambadur, and Anastasios Zouzias. A Randomized Algorithm for Approximating the Log Determinant of a Symmetric Positive Definite Matrix. *CoRR*, abs/1503.00374, 2015.

[26] P Oscar Boykin, Meera Sitharam, Mohamad Tarifi, and Pawel Wocjan. Real mutually unbiased bases. *arXiv preprint quant-ph/0502024*, 2005.

[27] Andrew M Bradley. H-matrix and block error tolerances. *arXiv preprint arXiv:1110.2807*, 2011.

[28] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[29] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[30] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[31] Peter W Buchen and Michael Kelly. The Maximum Entropy Distribution of an Asset inferred from Option Prices. *Journal of Financial and Quantitative Analysis*, 31(01):143–159, 1996.

[32] Paul Butterley and William Hall. Numerical evidence for the maximum number of mutually unbiased bases in dimension six. *Physics Letters A*, 369(1):5–8, 2007.

[33] Toon Calders, Asim Karim, Faisal Kamiran, Wasif Ali, and Xiangliang Zhang. Controlling attribute effect in linear regression. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 71–80. IEEE, 2013.

[34] Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010.

[35] Yanshuai Cao and David J Fleet. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

[36] A. Caticha. Entropic Inference and the Foundations of Physics (monograph commissioned by the 11th Brazilian Meeting on Bayesian Statistics–EBEB-2012, 2012.

[37] Krzysztof Chalupka, Christopher KI Williams, and Iain Murray. A framework for evaluating approximation methods for gaussian process regression. *The Journal of Machine Learning Research*, 14(1):333–350, 2013.

[38] Pieter Coulier, Hadi Pouransari, and Eric Darve. The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems. *arXiv preprint arXiv:1508.01835*, 2015.

[39] Kurt Cutajar, Michael Osborne, John Cunningham, and Maurizio Filippone. Preconditioning Kernel Matrices. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24*, 2016.

[40] Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[41] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic Metric Learning. In *Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 209–216, 2007.

[42] Timothy A Davis and Yifan Hu. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[43] Marc Peter Deisenroth and Jun Wei Ng. Distributed Gaussian Processes. *arXiv preprint arXiv:1502.02843*, 2015.

[44] Ashwini Deshpande. Affirmative action in india. In *Race and Inequality*, pages 77–90. Routledge, 2017.

[45] William Dieterich, Christina Mendoza, and Tim Brennan. Compas risk scales: Demonstrating accuracy equity and predictive parity. *Northpoint Inc*, 2016.

[46] Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*, pages 2791–2801, 2018.

[47] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580, 2018.

[48] Louis Dumont. *Homo hierarchicus: The caste system and its implications*. University of Chicago Press, 1980.

[49] Thomas Durt, Berthold-Georg Englert, Ingemar Bengtsson, and Karol Życzkowski. On mutually unbiased bases. *International journal of quantum information*, 8(04):535–640, 2010.

[50] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[51] Cynthia Dwork and Christina Ilvento. Group fairness under composition. *FATML*, 2018.

[52] Cynthia Dwork, Nicole Immorlica, Adam Tauman Kalai, and Max Leiserson. Decoupled classifiers for group-fair and efficient machine learning. In *Conference on Fairness, Accountability and Transparency*, pages 119–133, 2018.

[53] Jean-Pierre Eckmann and Elisha Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the national academy of sciences*, 99(9):5825–5829, 2002.

[54] Jane Elith, John R Leathwick, and Trevor Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008.

[55] Maurizio Filippone and Raphael Engler. Enabling Scalable Stochastic Gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE). In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11*, 2015.

[56] Jack Fitzsimons, Kurt Cutajar, Michael Osborne, Stephen Roberts, and Maurizio Filippone. Bayesian inference of log determinants. *arXiv preprint arXiv:1704.01445*, 2017.

[57] Jack Fitzsimons, Kurt Cutajar, Michael Osborne, Stephen Roberts, and Maurizio Filippone. Bayesian Inference of Log Determinants, 2017.

[58] Jack Fitzsimons, Diego Granziol, Kurt Cutajar, Michael Osborne, Maurizio Filippone, and Stephen Roberts. Entropic trace estimates for log determinants. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 323–338. Springer, 2017.

[59] Jack K. Fitzsimons, Michael A. Osborne, Stephen J. Roberts, and Joseph F. Fitzsimons. Improved Stochastic Trace Estimation using Mutually Unbiased Bases. *CoRR*, abs/1608.00117, 2016.

[60] Charless Fowlkes, Serge Belongie, and Jitendra Malik. Efficient spatiotemporal grouping using the nystrom method. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–231. IEEE, 2001.

[61] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[62] Pratik Gajane. On formalizing fairness in prediction with machine learning. *arXiv preprint arXiv:1710.03184*, 2017.

[63] Marc G Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of machine learning research*, 2(Dec):299–312, 2001.

[64] Semyon Gershgorin. Uber die Abgrenzung der Eigenwerte einer Matrix. *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7(3):749–754, 1931.

[65] Semyon Gershgorin. Uber die Abgrenzung der Eigenwerte einer Matrix. *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7(3):749–754, 1931.

[66] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[67] Adom Giffin, Carlo Cafaro, and Sean Alan Ali. Application of the Maximum Relative Entropy method to the Physics of Ferromagnetic Materials. *Physica A: Statistical Mechanics and its Applications*, 455:11 – 26, 2016.

[68] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[69] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. The Johns Hopkins University Press, 3rd edition, October 1996.

[70] Diego González, Sergio Davis, and Gonzalo Gutiérrez. Newtonian Dynamics from the Principle of Maximum Caliber. *Foundations of Physics*, 44(9):923–931, 2014.

[71] Robert B Gramacy. tgp: an r package for bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. *Journal of Statistical Software*, 19(9):6, 2007.

[72] Robert B Gramacy and Herbert KH Lee. Bayesian treed gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483), 2008.

[73] Diego Granziol and Stephen Roberts. An Information and Field Theoretic approach to the Grand Canonical Ensemble, 2017.

[74] Joseph Guinness and Ilse C. F. Ipsen. Efficient Computation of Gaussian Likelihoods for Stationary Markov Random Fields, 2015.

[75] Eldad Haber, Matthias Chung, and Felix Herrmann. An effective method for parameter estimation with pde constraints with multiple right-hand sides. *SIAM Journal on Optimization*, 22(3):739–757, 2012.

[76] Wolfgang Hackbusch and Steffen Börm. Data-sparse approximation by adaptive h^2-matrices. *Computing*, 69(1):1–35, 2002.

[77] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale log-determinant computation through stochastic chebyshev expansions. In *International Conference on Machine Learning*, pages 908–917, 2015.

[78] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale Log-Determinant computation through Stochastic Chebyshev Expansions. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015.

[79] Insu Han, Dmitry Malioutov, and Jinwoo Shin. Large-scale Log-Determinant computation through Stochastic Chebyshev Expansions. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015.

[80] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

[81] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[82] Trevor J Hastie and Robert J Tibshirani. Generalized additive models, volume 43 of monographs on statistics and applied probability, 1990.

[83] Philipp Hennig, Michael A. Osborne, and Mark Girolami. Probabilistic Numerics and Uncertainty in Computations. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.

[84] John Ben Hough, Manjunath Krishnapur, Yuval Peres, et al. *Zeros of Gaussian analytic functions and determinantal point processes*, volume 51. American Mathematical Soc., 2009.

[85] MF Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[86] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

[87] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

[88] Carl Jidling, Niklas Wahlström, Adrian Wills, and Thomas B Schön. Linearly constrained gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1215–1224, 2017.

[89] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[90] Matthew Joseph, Michael Kearns, Jamie Morgenstern, Seth Neel, and Aaron Roth. Rawlsian fairness for machine learning. *arXiv preprint arXiv:1610.09559*, 1(2), 2016.

[91] Faisal Kamiran and Toon Calders. Classifying without discriminating. In *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, pages 1–6. IEEE, 2009.

[92] Faisal Kamiran, Asim Karim, and Xiangliang Zhang. Decision theory for discrimination-aware classification. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 924–929. IEEE, 2012.

[93] Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 643–650. IEEE, 2011.

[94] Niki Kilbertus, Philip J Ball, Matt J Kusner, Adrian Weller, and Ricardo Silva. The sensitivity of counterfactual fairness to unmeasured confounding. *arXiv preprint arXiv:1907.01040*, 2019.

[95] Niki Kilbertus, Mateo Rojas Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. In *Advances in Neural Information Processing Systems*, pages 656–666, 2017.

[96] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[97] A Yu Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[98] Andreas Klappenecker and Martin Rötteler. Constructions of mutually unbiased bases. In *Finite fields and applications*, pages 137–144. Springer, 2004.

[99] Jon Kleinberg, Jens Ludwig, Sendhil Mullainathan, and Ashesh Rambachan. Algorithmic fairness. In *Aea papers and proceedings*, volume 108, pages 22–27, 2018.

[100] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[101] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, pages 4066–4076, 2017.

[102] Frédéric Lavancier, Jesper Møller, and Ege Rubak. Determinantal point process models and statistical inference. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(4):853–877, 2015.

[103] Judith Y Li, Sivaram Ambikasaran, Eric F Darve, and Peter K Kitanidis. A kalman filter powered by $\mathcal{H}^2$-matrices for quasi-continuous data assimilation problems. *arXiv preprint arXiv:1404.3816*, 2014.

[104] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498, 2011.

[105] Binh Thanh Luong, Salvatore Ruggieri, and Franco Turini. k-nn as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 502–510. ACM, 2011.

[106] Odile Macchi. The Coincidence Approach to Stochastic point processes. *Advances in Applied Probability*, 7:83–122, 1975.

[107] David J. C. Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, first edition edition, June 2003.

[108] Leslie McCall. The complexity of intersectionality. In *Intersectionality and Beyond*, pages 65–92. Routledge-Cavendish, 2008.

[109] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[110] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondvrej vCertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, vStvepán Rouvcka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[111] Ali Mohammad-Djafari. A matlab program to calculate the maximum entropy distributions. In *Maximum Entropy and Bayesian Methods*, pages 221–233. Springer, 1992.

[112] Nikola Mrkvsić. Kernel structure discovery for Gaussian process classification. Master's thesis, Computer Laboratory, University of Cambridge, June 2014.

[113] Cassio Neri and Lorenz Schneider. Maximum Entropy Distributions inferred from Option Portfolios on an Asset. *Finance and Stochastics*, 16(2):293–318, 2012.

[114] Jun Wei Ng and Marc Peter Deisenroth. Hierarchical mixture-of-experts model for large-scale gaussian process regression. *arXiv preprint arXiv:1412.3078*, 2014.

[115] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[116] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

[117] A. O'Hagan. Bayes-Hermite Quadrature. *Journal of Statistical Planning and Inference*, 29:245–260, 1991.

[118] Anthony O'Hagan. Bayes–hermite quadrature. *Journal of statistical planning and inference*, 29(3):245–260, 1991.

[119] Michael Osborne, Roman Garnett, Zoubin Ghahramani, David K Duvenaud, Stephen J Roberts, and Carl E Rasmussen. Active learning of model evidence

using bayesian quadrature. In *Advances in neural information processing systems*, pages 46–54, 2012.

[120] R Kelley Pace and James P LeSage. Chebyshev approximation of log-determinants of spatial weight matrices. *Computational Statistics & Data Analysis*, 45(2):179–196, 2004.

[121] Christopher C Paige. Computational Variants of the Lanczos method for the Eigenproblem. *IMA Journal of Applied Mathematics*, 10(3):373–381, 1972.

[122] Karl Pearson. *The life, letters and labours of Francis Galton*, volume 3. CUP Archive, 1930.

[123] Wei Peng and Hongxia Wang. Large-scale Log-Determinant Computation via Weighted L_2 Polynomial Approximation with Prior Distribution of Eigenvalues. In *International Conference on High Performance Computing and Applications*, pages 120–125. Springer, 2015.

[124] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, pages 5680–5689, 2017.

[125] Steve Pressé, Kingshuk Ghosh, Julian Lee, and Ken A. Dill. Principles of Maximum Entropy and Maximum Caliber in Statistical Physics. *Reviews of Modern Physics*, 85:1115–1141, Jul 2013.

[126] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A Unifying View of Sparse Approximate Gaussian Process Regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[127] Edward Raff, Jared Sylvester, and Steven Mills. Fair forests: Regularized tree induction to minimize model bias. *arXiv preprint arXiv:1712.08197*, 2017.

[128] Edward Raff, Jared Sylvester, and Steven Mills. Fair forests: Regularized tree induction to minimize model bias. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 243–250. ACM, 2018.

[129] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[130] Carl E. Rasmussen and Christopher Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[131] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[132] Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. In *Advances in Neural Information Processing Systems 15, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada*, pages 489–496, 2002.

[133] Lior Rokach and Oded Z Maimon. *Data mining with decision trees: theory and applications*, volume 69. World scientific, 2008.

[134] Havard Rue and Leonhard Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[135] Håvard Rue and Leonhard Held. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 2005.

[136] Håvard Rue and Leonhard Held. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London, 2005.

[137] Håvard Rue, Sara Martino, and Nicolas Chopin. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009.

[138] Yunus Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

[139] Arvind K Saibaba, Sivaram Ambikasaran, J Yue Li, Peter K Kitanidis, and Eric F Darve. Application of hierarchical matrices to linear inverse problems in geostatistics. *Oil and Gas Science and Technology-Revue de l'IFP-Institut Francais du Petrole*, 67(5):857, 2012.

[140] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[141] Julian Schwinger. Unitary operator bases. *Proceedings of the National Academy of Sciences*, 46(4):570–579, 1960.

[142] Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

[143] John Shore and Rodney Johnson. Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-Entropy. *IEEE Transactions on information theory*, 26(1):26–37, 1980.

[144] RN Silver and H Röder. Calculation of densities of states and spectral functions by chebyshev recursion and maximum entropy. *Physical Review E*, 56(4):4822, 1997.

[145] Jack W Silverstein. Eigenvalues and Eigenvectors of Large Dimensional Sample Covariance Matrices. *Contemporary Mathematics*, 50:153–159, 1986.

[146] Alex J Smola and Peter Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*. Citeseer, 2001.

[147] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2005.

[148] Jeffrey M Stanton. Galton, pearson, and the peas: A brief history of linear regression for statistics instructors. *Journal of Statistics Education*, 9(3):1–16, 2001.

[149] Michael L. Stein, Jie Chen, and Mihai Anitescu. Stochastic Approximation of Score functions for Gaussian processes. *The Annals of Applied Statistics*, 7(2):1162–1191, 2013.

[150] Michael L Stein, Jie Chen, Mihai Anitescu, et al. Stochastic approximation of score functions for gaussian processes. *The Annals of Applied Statistics*, 7(2):1162–1191, 2013.

[151] Volker Tresp. A bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

[152] Charalampos E Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 608–617. IEEE, 2008.

[153] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast Estimation of tr (f (a)) via Stochastic Lanczos Quadrature. 2016.

[154] Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast Estimation of tr (f (a)) via Stochastic Lanczos Quadrature. 2016.

[155] United States. Executive Office of the President and John Podesta. *Big data: Seizing opportunities, preserving values*. White House, Executive Office of the President, 2014.

[156] Tristan van Leeuwen, Aleksandr Y Aravkin, and Felix J Herrmann. Seismic waveform inversion by stochastic optimization. *International Journal of Geophysics*, 2011, 2011.

[157] Jon Wakefield. *Bayesian and frequentist regression methods*. Springer Science & Business Media, 2013.

[158] Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.

[159] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[160] Andrew J. Wathen and Shengxin Zhu. On Spectral Distribution of Kernel Matrices related to Radial Basis functions. *Numerical Algorithms*, 70(4):709–726, 2015.

[161] Thomas E Weisskopf. Affirmative action in the United States and India. *A Comparative Perspective, London and New York: Routledge*, 2004.

[162] Hermann Weyl. Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen (mit einer Anwendung auf die Theorie der Hohlraumstrahlung). *Mathematische Annalen*, 71(4):441–479, 1912.

[163] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems*, number EPFL-CONF-161322, pages 682–688, 2001.

[164] Christopher KI Williams and Carl Edward Rasmussen. Gaussian Processes for Machine Learning. *the MIT Press*, 2(3):4, 2006.

[165] Wolfram Research Inc. Mathematica.

[166] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[167] 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. International World Wide Web Conferences Steering Committee, 2017.

[168] Muhammad Bilal Zafar, Isabel Valera, Manuel Rodriguez, Krishna Gummadi, and Adrian Weller. From parity to preference-based notions of fair-

ness in classification. In *Advances in Neural Information Processing Systems*, pages 229–239, 2017.

[169] Yunong Zhang and William E Leithead. Approximate Implementation of the logarithm of the Matrix Determinant in Gaussian process Regression. *Journal of Statistical Computation and Simulation*, 77(4):329–348, 2007.

[170] Zhikuan Zhao, Vedran Dunjko, Jack K Fitzsimons, Patrick Rebentrost, and Joseph F Fitzsimons. A note on state preparation for quantum machine learning. *arXiv preprint arXiv:1804.00281*, 2018.

[171] Zhikuan Zhao, Jack K Fitzsimons, and Joseph F Fitzsimons. Quantum assisted gaussian process regression. *arXiv preprint arXiv:1512.03929*, 2015.

[172] Zhikuan Zhao, Jack K Fitzsimons, Michael A Osborne, Stephen J Roberts, and Joseph F Fitzsimons. Quantum algorithms for training gaussian processes. *arXiv preprint arXiv:1803.10520*, 2018.

[173] Zhikuan Zhao, Jack K Fitzsimons, Michael A Osborne, Stephen J Roberts, and Joseph F Fitzsimons. Quantum algorithms for training gaussian processes. *arXiv preprint arXiv:1803.10520*, 2018.