Bayesian Machine Learning for Financial Modeling

Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften

vorgelegt beim Fachbereich Informatik und Mathematik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

von
Rajbir Singh Nirwan
aus Chahal Khurd

Frankfurt (2020)
(D 30)

vom Fachbereich Informatik und Mathematik der
Johann Wolfgang Goethe-Universität als Dissertation angenommen.

Dekan: Prof. Dr. Lars Hedrich

Gutachter: ____

____

Datum der Disputation: ____

GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN

Bayesian Machine Learning for Financial Modeling

Rajbir Singh Nirwan

Department of Computer Science and Mathematics
Goethe-University Frankfurt am Main

This dissertation is submitted for the degree of
Doctor of Natural Sciences

September 2020

Abstract

Machine Learning (ML) is so pervasive in our todays life that we don't even realise that, more often than expected, we are using systems based on it. It is also evolving faster than ever before. When deploying ML systems that make decisions on their own, we need to think about their ignorance of our uncertain world. The uncertainty might arise due to scarcity of the data, the bias of the data or even a mismatch between the real world and the ML-model. Given all these uncertainties, we need to think about how to build systems that are not totally ignorant thereof. Bayesian ML can to some extent deal with these problems. The specification of the model using probabilities provides a convenient way to quantify uncertainties, which can then be included in the decision making process.

In this thesis, we introduce the Bayesian ansatz to modeling and apply Bayesian ML models in finance and economics. Especially, we will dig deeper into Gaussian processes (GP) and Gaussian process latent variable model (GPLVM). Applied to the returns of several assets, GPLVM provides the covariance structure and also a latent space embedding thereof. Several financial applications can be build upon the output of the GPLVM. To demonstrate this, we build an automated asset allocation system, a predictor for missing asset prices and identify other structure in financial data.

It turns out that the GPLVM exhibits a rotational symmetry in the latent space, which makes it harder to fit. Our second publication reports, how to deal with that symmetry. We propose another parameterization of the model using Householder transformations, by which the symmetry is broken. Bayesian models are changed by reparameterization, if the prior is not changed accordingly. We provide the correct prior distribution of the new parameters, such that the model, i.e. the data density, is not changed under the reparameterization. After applying the reparametrization on Bayesian PCA, we show that the symmetry of nonlinear models can also be broken in the same way.

In our last project, we propose a new method for matching quantile observations, which uses order statistics. The use of order statistics as the likelihood, instead of a Gaussian likelihoood, has several advantages. We compare these two models and highlight their advantages and disadvantages. To demonstrate our method, we fit quantiled salary data of several European countries. Given several candidate models for the fit, our method also provides a metric to choose the best option.

We hope that this thesis illustrates some benefits of Bayesian modeling (especially Gaussian processes) in finance and economics and its usage when uncertainties are to be quantified.

Zusammenfassung (Summary in German)

Die vorliegende Arbeit beschäftigt sich mit bayesianischer Statistik und ihrer Anwendung im Bereich Finanzen und Ökonomie.

Es wurden drei verschiedene Themen behandelt, welche bereits veröffentlicht sind oder in einer veröffentlichungswürdigen Form vorliegen. Diese Dissertation ist eine Erweiterung der Veröffentlichungen "Applications of Gaussian process Latent Variable Models in Finance" (Nirwan and Bertschinger, 2019b) und "Rotation Invariant Householder Parameterization for Bayesian PCA" (Nirwan and Bertschinger, 2019a) und der noch nicht veröffentlichten Arbeit "Bayesian Quantile Matching Estimation" (Nirwan and Bertschinger, 2020). Die Erweiterung umfasst eine detaillierte Einführung in die Themen und Beschreibung der Modelle und Experimente.

Nach der allgemeinen Einführung in das maschinelle Lernen und der Wichtigkeit der Einschätzung von Unsicherheiten in Kapitel 1, geben wir einen Überblick in die bayesianische Statistik in Kapitel 2. Der Überblick vergleicht das klassische Vorgehen (frequentistisch) mit dem bayesianischen Ansatz anhand der linearen Regression (LR). Das mathematisch sehr simple LR Modell ist gut zu interpretieren und ist analytisch lösbar, kann aber aufgrund seiner mangelnden Flexibilität die Struktur von komplexen Datensätzen nicht erfassen. In diesem Kapitel zeigen wir den Vorteil des bayesianischen Ansatzes gegenüber des klassischen Ansatzes. Dieser liegt in der Abschätzung von Unsicherheiten in den gelernten Parameterwerten. Die Unsicherheit kann quantifiziert werden, indem man Modelle durch Wahrscheinlichkeiten beschreibt (Bishop, 2006). Das Resultat ist dann nicht ein Wert für einen gelernten Parameter, sondern eine Verteilung über mögliche Werte (der Posterior).

Wie bereits erwähnt, ist die lineare Regression zwar analytisch lösbar, ihr mangelt es aber an Flexibilität. Flexiblere Modelle können mehr Struktur erfassen, sind aber dafür nicht mehr analytisch lösbar und müssen approximativ gelöst werden. Es gibt zwei große Klassen der Approximationen einer Verteilung (in unserem Fall, des Posteriors). Eine davon, Markov Chain Monte Carlo (MCMC), approximiert den Posterior durch Samples. Die andere, Variational Bayes (VB), approximiert den Posterior durch eine einfachere analytische Verteilung. In Kapitel 3 stellen wir beide Klassen vor und gehen tiefer auf Hamiltonian Monte Carlo (HMC, ein MCMC Verfahren) (Betancourt, 2017) und Variational Inference (VI, ein VB Verfahren) (Bishop, 2006; MacKay, 2002) ein. Wir benutzen sowohl HMC, als auch VI in den Experimenten. Glücklicherweise muss man diese Methoden (HMC und VI) nicht selbst programmieren. Es gibt probabilistische Programmiersprachen, die die Im-

plementierung der bayesianischen Modelle vereinfachen. In dieser Arbeit haben wir Stan (Carpenter et al., 2017) verwendet, welches wir in Kapitel 3 erläutern. Stan liefert eine Implementierung von HMC und VI, das man Out-of-the-box verwenden kann. Somit muss man sich nicht mehr um die Solver kümmern, sondern kann sich voll auf die Modellierung konzentrieren.

Kapitel 4 und 5 sind Einführungen in die bayesianische Machine Learning Modelle, welche wir in den Veröffentlichungen verwenden. Kapitel 4 erklärt detailliert die Funktionsweise der Gaußschen Prozesse (Rasmussen and Williams, 2005). Gaußsche Prozesse sind Wahrscheinlichkeitsverteilungen von Funktionen und werden bei der Interpolation und Extrapolation von Daten verwendet. Genauso wie eine Gauß-Verteilung sind Gaußsche Prozesse durch den Erwartungswert und einer Kovarianzfunktion eindeutig bestimmt. Die Kovarianzfunktion bestimmt die Eigenschaften (Stetigkeit, Differenzierbarkeit, ...) der Funktionssamples aus dem Prozess. Diese wird auch in unseren Analysen eine wichtige Rolle einnehmen. Kapitel 5 behandelt Latent Variable Models (LVM) (Bishop, 2006). Diese werden häufig in Unsupervised-Learning benutzt, wenn Daten ohne Labels vorliegen und man an den latenten Variablen interessiert ist, welche die Daten generieren. Die Kombination von GPs und LVMs ergibt das Gaussian process latent variable model (GPLVM), eingeführt in Lawrence (2005). Wir benutzen GPLVM, um Struktur in nicht gelabelten Finanzdaten zu erkennen.

Kapitel 6 bis Kapitel 10 erklären detaillierter die Veröffentlichungen. Nach der Einführung in die Themen werden die Erweiterungen der Modelle und die Experimente beschrieben.

Gaussian process latent variable models in Finance

Kapitel 6 führt die Finanz-Modelle ein. Unter anderem werden das Capital Asset Pricing Model (CAPM, Sharpe (1964)) und Arbitrage theory of capital asset pricing (APT, Ross (1976)) eingeführt. Diese beschreiben die Abhängigkeit der Rendite eines Assets von anderen latenten Faktoren. Diese Faktoren sind nicht beobachtbar, sondern werden von den bereits beobachteten Daten abgeleitet. Es wird eine lineare Abhängigkeit der Faktoren und der Rendite vorausgesetzt. Das Modell ist bekannt unter Faktor-Modell (Everett, 1984).

Die erste Veröffentlichung (Nirwan and Bertschinger, 2019b) verallgemeinert diese Modelle durch die GPLVMs. Kapitel 7 zeigt, dass durch das Benutzen eines linearen Kernels, GPLVM auf APT zurückgeführt werden kann. In den Experimenten schauen wir uns die Aktienpreise der Unternehmen in den S&P500 an. Die Zeitreihen der Preise für verschiedene Unternehmen werden in einer Matrix zusammengefasst und daraus die Returns (Änderung des Preises von einem Tag auf den nächsten) berechnet. Die Matrix der Returns bildet die Basis der Experimente. Wir zeigen, dass das GPLVM mit nicht-linearen Kernels in der Lage ist, mehr Struktur in den Daten zu erfassen als mit linearem Kernel, welches äquivalent zu APT ist. Die Anwendung der GPLVMs gibt uns die Latent Space Repräsentation $\boldsymbol{X}$ der Assets. Außerdem liefert das Modell auch die Kovarianz Matrix $\boldsymbol{K}$ der Assets. Wir benutzen sowohl $\boldsymbol{X}$ als auch $\boldsymbol{K}$ für weitere Analysen.

Es gibt viele verschiedene Finanzanwendungen, in denen man das GPLVM benutzen kann.

Wir stellen drei davon vor.
Portfolio Allokation: Hier benutzen wir die Kovarianz der Assets $\boldsymbol{K}$ um ein risikoarmes Portfolio (minimal risk portfolio) zu erstellen. Wir simulieren die Performance auf Basis historischer Werte und vergleichen das Risiko unseres Portfolios mit anderen Kovarianz-Schätzern. Dabei benutzen wir die Daten der Unternehmen in den S&P500 von 2002 bis 2018. Die aufgestellten Portfolios durch das GPLVM mit nicht-linearen Kernels ergaben das geringste Risiko.
Bestimmung der fehlenden Werte: Falls ein Asset an einem Tag nicht gekauft wurde, gibt es keinen Wert (Kaufpreis) dafür. Als Proxy kann der Wert des letzten Tags genommen werden oder ein Durchschnitt der Werte der letzten Tage. Der konstruierte Latent Space $\boldsymbol{X}$ durch das GPLVM gibt uns eine andere Möglichkeit, den fehlenden Wert zu bestimmen. Gegeben $\boldsymbol{X}$, können wir einen standard GP an dem Tag trainieren, an dem der Wert fehlt und haben somit einen Schätzer für fehlende Werte, der nicht nur die Information aus der Historie eines Assets, sondern auch die Korrelation des Assets mit anderen Assets mitberücksichtigt.
Latent Space Repräsentation: Für stationäre Kernels hat der Abstand zweier Assets in den Latent Space einen direkten Einfluss auf deren Korrelation. Assets, die nahe beieinander sind, haben eine größere Korrelation als Assets, die weiter weg voneinander liegen. Dies führt zu einer Gruppierung der Assets basierend auf ihrem Teilsektor. Diese Struktur im Latent Space kann genutzt werden, um Portfolios mit dekorrelierten Assets zu bilden. Zum Beispiel kann man aus N Assets M viele in das Portfolio aufnehmen, die den paarweisen Abstand maximieren. Man kann auch ein Portfolio aus der konvexen Hülle konstruieren.

Rotation invariant Householder parameterization for Bayesian PCA

GPLVM mit linearen und stationären Kernels hat das Problem, dass der Latent Space nicht eindeutig festgelegt ist, sondern eine Rotationssymmetrie aufweist. Diese Symmetrie erschwert nicht nur die Interpretation des Latent Space's, sondern auch dessen Exploration beim Samplen (z. B.: durch Hamiltonian Monte Carlo). In Kapitel 8 schlagen wir eine Umparametrisierung des Modells vor, die das Modell nicht ändert (erzeugt dieselbe Datendichte wie das ursprüngliche Modell), aber trotzdem die Rotationssymmetrie bricht. Nirwan and Bertschinger (2019a) beinhaltet die Veröffentlichung.
Bei einem probabilistischen Modell muss man beachten, dass eine Umparametrisierung (Substituion) auch einer Korrektur des Priors bedarf, damit das Modell gleich bleibt. Kennt man aber die richtige Verteilung der neuen Parameter, braucht man die Korrektur nicht zu berechnen. Für einen linearen Kernel entspricht das GPLVM der Principle Component Analysis (PCA, Tipping and Bishop (1999)), welches wir als Beispiel für unsere Umparametrisierung nehmen. PCA bildet die beobachteten Daten $\boldsymbol{Y}$ auf eine niederdimensionale Mannigfaltigkeit ab. Die probabilistische Variante kann man sich als ein generatives Modell vorstellen, welches die niederdimensionalen Daten $\boldsymbol{X}$ über eine lineare Abbildung $\boldsymbol{W}$ nach $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X} + \boldsymbol{\epsilon}$ abbildet. Für einen Gauß Prior auf $\boldsymbol{X}$ kann man $\boldsymbol{X}$

analytisch raus integrieren. Die marginale Verteilung (marginal likelihood) $p(\boldsymbol{Y}|\boldsymbol{W})$ hängt dann von dem äußeren Produkt $\boldsymbol{W}\boldsymbol{W}^T$ ab. Eine Rotation von $\boldsymbol{W}$ ändert $p(\boldsymbol{Y}|\boldsymbol{W})$ nicht. Deshalb ist das Modell rotationsinvariant. Wir brechen die Rotationssymmetrie, indem wir $\boldsymbol{W}$ durch Singulärwertzerlegung in $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ und $\boldsymbol{V}$ zerlegen ($\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$) und statt $\boldsymbol{W}$ als Parameter, $\boldsymbol{U}$ und $\boldsymbol{\Sigma}$ als Parameter verwenden. $\boldsymbol{V}$ kann ignoriert werden, da das äußere Produkt $\boldsymbol{W}\boldsymbol{W}^T$ nicht von $\boldsymbol{V}$ abhängt[1]. $\boldsymbol{W}$ in der ursprünglichen Parametrisierung hat einen Gauß Prior, sodass $\boldsymbol{W}\boldsymbol{W}^T$ eine Wishart Verteilung aufweist. In Kapitel 8 berechnen wir die Prior Verteilung auf $\boldsymbol{U}$ und $\boldsymbol{\Sigma}$, sodass $\boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T$ auch einer Wishart Verteilung entspricht. Die Verteilungen auf $\boldsymbol{U}$ und $\boldsymbol{\Sigma}$, die eine Wishart Verteilung auf $\boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T$ implizieren, lassen das Modell invariant unter der Umparametrisierung. Außer der Brechung der Rotationssymmetrie, kommt die neue Parametrisierung mit vielen anderen Vorteilen:

Weniger Parameter: Da $\boldsymbol{U}$ eine orthogonale Matrix ist, hat sie nur $DQ - \frac{1}{2}Q(Q+1)$ viele Freiheitsgrade. Wir benutzen Householder Transformationen um $\boldsymbol{U}$ zu parametrisieren, welche die Anzahl der freien Parameter nicht erhöhen. Zusätzlich kommen Q Freiheitsgrade durch die Singulärwerte $\boldsymbol{\Sigma}$ dazu. Im Gegensatz dazu hat $\boldsymbol{W}$ DQ viele Freiheitsgrade.

Recheneffizienz: Da wir die richtige Verteilung auf den neuen Parametern kennen, brauchen wir die Jacobi-Determinante nicht zu berechnen. Andere Vorschläge, wie zum Beispiel die givens Rotations (Shepard et al., 2014), brauchen eine Jacobi-Korrektur für die DQ Parameter, welches eine Komplexität der Ordnung $\mathcal{O}(D^3Q^3)$ hat.

Interpretierbarkeit: Wir zerlegen die Abbildung $\boldsymbol{W}$ über eine Singulärwertzerlegung in $\boldsymbol{U}$ und $\boldsymbol{\Sigma}$. $\boldsymbol{U}$ ist eine Rotation des Datenraumes und $\boldsymbol{\Sigma}$ sind die Singulärwerte. Diese Interpretation erlaubt es, das Wissen über die Rotation des Datenraumes oder die Ordnung der Singulärwerte in deren Prior einzubinden. Zum Beispiel kann man einen Sparsity-Prior auf die Singulärwerte legen, wenn man nur wenige latente Dimensionen erwartet. Im Gegensatz dazu ist $\boldsymbol{W}$ nicht so gut interpretierbar.

Erweiterung auf nicht-lineare Modelle: In den Experimenten zeigen wir, dass wir auch die Rotation in den Latent Space der nicht-linearen Modelle brechen können. Als Beispiel nehmen wir wieder das GPLVM mit dem Gauß Kernel.

Bayesian quantile matching estimation

In dem letzten Projekt schlagen wir eine neue Methode für das Lernen von Wahrscheinlichkeitsverteilungen aus Quantildaten vor. Unsere Methode ist eine Alternative zu der bisher meist genutzten Mean Squared Error (MSE) Minimierung. In der MSE-Minimierung wird eine kumulative Dichtefunktion (CDF) an die beobachteten Quantildaten gefittet. Wir zeigen, dass dabei die Annahme für den Fehler um die CDF nicht gerechtfertigt ist und die Ränder der echten Verteilung nicht akkurat abgebildet werden. Unsere Methode benutzt die Ordnungsstatistik der Quantilwerte als Fehlermaß, welches in diesem Fall ein besseres Maß als die quadratische Abweichung von der CDF ist.

Wir führen die Ordnungsstatistik in Kapitel 9 ein und beschreiben, im darauf aufbauenden Kapitel 10, unsere Methode für das Lernen von Verteilungen aus Quantildaten. Die

[1] $\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T$

vorgeschlagene Methode hat sehr viele Vorteile gegenüber der MSE-Minimierung, welche im Folgenden aufgelistet sind. Wir zeigen, sowohl mit synthetischen als auch mit echten Daten, dass für das Fitten von Quantildaten unsere Methode besser geeignet ist.
Abbildung der Ränder der Verteilung: Beobachtete Quantildaten haben eine höhere Unsicherheit an den Rändern, weil empirisch weniger Samples aus den Rändern zur Verfügung stehen, um diese Abzuschätzen. Das Fehlermaß, welches wir über die Ordnungsstatistik bekommen, bildet diese Unsicherheit korrekt ab und ist somit besser geeignet für die Abschätzung der Ränder.
Abschätzung der Parameterunsicherheit: Der bayesianische Ansatz liefert eine Unsicherheit der Modellparameter in Form der Posterior Verteilung. Unsere Experimente mit synthetischen Daten zeigen, dass der bayesianische Ansatz mit MSE-Minimierung die echte Unsicherheit vollkommen unterschätzt. Im Gegensatz dazu kann der bayesianische Ansatz mit unserer Methode die Unsicherheiten deutlich besser abschätzen.
Modell Auswahl: Wir zeigen, wie man mit unserer Methode aus mehreren Verteilungen die Beste auswählt. Dazu analysieren wir Gehaltsdaten von mehreren europäischen Ländern aus 2016. Gegeben sind die 25%, 50% und 75% Quantile der Gehaltsverteilungen. Wir zeigen, dass die Weibull-, Log-normal- und Gamma-Verteilung die besten Verteilungsannahmen für diese Art von Daten sind.

Zum Schluss beinhaltet Kapitel 11 das Fazit und fasst noch einmal die Dissertation zusammen.

Contents

Abstract **v**

1 Introduction **1**
1.1 Why uncertainty matters . . . 2
1.2 Frequentist vs Bayesian . . . 3
1.3 Structure of the thesis . . . 4

2 Bayesian modeling **5**
2.1 Introduction . . . 5
2.2 Linear regression . . . 6
2.2.1 Frequentists linear regression . . . 7
2.2.2 Bayesian linear regression . . . 10
2.2.3 Summary . . . 15

3 Approximations to the posterior **17**
3.1 Sampling . . . 18
3.1.1 Monte Carlo sampling . . . 18
3.1.2 Markov chain Monte Carlo (MCMC) . . . 20
3.1.3 Hamiltonian Monte Carlo . . . 20
3.2 Variational inference . . . 22
3.3 Probabilistic programming . . . 25
3.4 Summary . . . 25

4 Gaussian processes **29**
4.1 Gaussian process as a limit to Gaussian distribution . . . 29
4.2 Gaussian process regression . . . 31
4.2.1 Weight space view . . . 32
4.2.2 Function space view . . . 33
4.2.3 Mercer's Theorem . . . 34
4.3 Kernel function and its Fourier transformation . . . 36
4.4 Computational complexity and approximations . . . 37

5 Bayesian latent variable models **41**
5.1 Introduction . . . 41
5.2 Principle Component Analysis . . . 43
5.2.1 Classical PCA . . . 43
5.2.2 Probabilistic PCA . . . 43
5.2.3 Connection of classical PCA and probabilistic PCA . . . 44
5.2.4 Bayesian PCA . . . 44
5.3 Gaussian process Latent Variable Models . . . 45

6 Finance **47**
6.1 Latent space models in finance . . . 47
6.1.1 Capital Asset Pricing Model (CAPM) . . . 47
6.1.2 Arbitrage theory of capital asset pricing . . . 48
6.1.3 Fama-French three-factor model . . . 49
6.2 Modern Portfolio Theory . . . 50
6.3 Estimation of covariance matrices . . . 50

7 Gaussian process latent variable models in finance **53**
7.1 Nonlinear extension of APT using GPLVMs . . . 53
7.2 Modeling and data collection . . . 54
7.2.1 Model comparison . . . 56
7.3 Experiments . . . 58
7.3.1 Portfolio allocation . . . 58
7.3.2 Fill in missing values . . . 60
7.3.3 Interpretation of the latent space . . . 61

8 Rotation invariant Householder parameterization for Bayesian PCA **65**
8.1 Problem definition . . . 65
8.2 Background . . . 66
8.2.1 Singular Value Decomposition . . . 67
8.2.2 Random Matrix Theory . . . 67
8.2.3 Householder transformations . . . 69
8.3 Unique PPCA . . . 72
8.3.1 Implementation . . . 72
8.3.2 Model comparison . . . 74
8.3.3 Computational complexity and runtime analysis . . . 77
8.3.4 Problems . . . 78
8.3.5 Interpretation of the parameters and other than Gaussian priors . . . 78
8.4 Extension to nonlinear models . . . 79

9 Order Statistics **83**
9.1 Order Statistics of a Uniform distribution . . . 83
9.2 Generalization to non-Uniform distributions . . . 84
9.3 Joint distribution of Order Statistics . . . 85

10 Bayesian quantile matching estimation 87
10.1 Problem definition . . . 87
10.2 Fit a non-Uniform distribution given quantile information . . . 89
10.2.1 Joint distribution of observed quantile values . . . 89
10.2.2 Generative model . . . 90
10.2.3 Model selection . . . 90
10.3 CDF regression model . . . 90
10.4 Experiments . . . 91
10.4.1 Bayesian quantile matching estimation . . . 91
10.4.2 Dependency on N . . . 92
10.4.3 Robustness to change of a single point in the sample . . . 93
10.4.4 Penalty for OS and CDF-fit . . . 95
10.5 Matching salary data of different countries . . . 96

11 Summary and conclusion 101
11.1 Gaussian process latent variable models in finance . . . 102
11.2 Rotation invariant Householder parameterization for Bayesian PCA . . . 102
11.3 Bayesian quantile matching estimation . . . 103

A Appendix 111
A.1 Normalization Constant . . . 111

Chapter 1

Introduction

The field of machine learning is evolving faster than ever before. The methods that were only applied to toy data are now being deployed in real life settings. Machine learning will profoundly change every aspect of our life in the next decades. It starts from mobility, where we will have self-driving cars and ends with medical treatments, where robots will take over critical interventions.

Machine learning systems are built to automatically track people's health to warn them before something bad happens (personalized medicine). If diseases are detected early they can be treated in a state when it is easier for the patient and also for the doctor. A good example of such an early warning system is the TREWScore for septic shocks (Henry et al., 2015).

Machine learning is not just applied in medicine, but also in many other areas as Neuroscience (understanding the brain, Chai et al. (2017)), Engineering (robotic and optimal control, Kober et al. (2013)), Linguistics (natural language processing, Vaswani et al. (2017)), Cognitive Science and Psychology (how learning takes place in humans, Bassett and Sporns (2017); Chai et al. (2017)), Economy (decision and game theory), Biology (decoding the human genome) and many more fields. Also the financial sector is already changing. Machine learning helps us to spot patterns that humans cannot spot. It might be fraud detection (Awoyemi et al., 2017), credit risk assessment (Bao et al., 2019) or just an agent giving advice on how to invest your money and what assets to buy. So, machine learning has been studied and used in many different fields, but the underlying methods are all coming from basic mathematics/statistics and are independent of the field they are used in. Machine Learning is an interdisciplinary field.

Due to the high success in many fields people are starting to allow machines to make automated decisions without active surveillance. With this, questions about safety are arising, and the demand of AI systems being aware of their uncertainty is increasing.

One of the challenges we have to deal with is to distinguish between when the model really knows something because it detected a pattern and when the model is just guessing at random. If we only receive a "yes" or "no" or a single number from the machine, we can never distinguish between, when the model guesses or when it makes true predictions based on the information it picked up. Fortunately, there is a framework that helps to distinguish exactly between these two possibilities. This framework which will be very important for critical systems is machine learning as probabilistic modeling. Uncertainties can be expressed via probabilities. Probabilistic models therefore are very well suited to incorporate those uncertainties in the input as well as to express believability of their predictions (Ghahramani, 2015).

1.1 Why uncertainty matters

Despite the negative touch associated with uncertainty, it is very important to know as much of the unknowns as possible. Even though we would like to know things with certainty, there are many things, where we cannot (e.g. unknown unknowns). There might be intrinsic factors of a problem that we cannot control that lead to uncertainty in modeling as well as decision making.

Uncertainty can arise in many different ways: e.g. if we simply do not have enough data to pinpoint a model (scarcity of the training data), if the collected data do not represent the full data distribution (biased training data). Also a mismatch between the model we choose and the actual model (the real world) that generated the data can lead to wrong inferences. The effect of the first two points (Scarcity[1] and the bias of training data[2]) can be mitigated to some extent by using probabilities to encode these uncertainties and building probabilistic models. Also the last point (mismatch between the model and the real world) is an ongoing research topic[3] and not fully solved, if ever solvable.

The paragraph above stated the problems that we have during the modeling phase. However, after modeling there is one more phase, the decision making phase, where based on the modeling and the predictions a decision has to be made. Here also another kind of uncertainty arises, which is the uncertainty related to not knowing the actual objective/utility function of the person the decision is actually affecting. The quantification of uncertainty in these settings is very challenging and we most certainly cannot quantify all of them. That is also not the goal of modeling in general[4]. However, we can still try to model as much uncertainty as we can and hope for the best.

So, if we want machine learning systems to make decisions for us we should, whenever possible, use models that can capture most of the above stated problems. Especially when the system is not just making predictions and assisting other people to make decisions but makes decisions by itself based on these predictions. Therefore, it is really essential that

[1] If not much data is available the posterior parameter distribution will be broader than in the presence of a lot of data, where the posterior distribution will become narrower reflecting the reduced uncertainty of the parameter estimate.

[2] The bias can be coped with to some degree by using informative priors. But also that is limited in its scope and introduces other kind of biases.

[3] See (Bishop, 2006) on Bayesian model averaging (BMA) and (Clyde and Iversen, 2013) on Bayesian model averaging in the “M-Open” framework, where goal is to come up with with a reasonably good description (combination of models) of the data while still assuming that the “true” data generating model is outside the space of models to be combined.

[4] Note that also intelligent people make decisions by reasoning despite scarcity of data, despite the bias of the data (not to forget: their own, sometimes very strong, biases) and despite not knowing what peoples objective is, who are affected by their decisions.

systems know when they “don't know”, i.e. they are aware of the ignorance/uncertainty in their ability to predict. Probabilities provide a nice way to do exactly that. Using the probabilistic framework in machine learning is known as Bayesian machine learning. In some aspects, it is different from the classical frequentists approach, even though in some settings they provide the same result.

1.2 Frequentist vs Bayesian

In a parametric model, for example, frequentists and Bayesians have the same setup. Both assume that there is an underlying function $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ that maps the input $\boldsymbol{x}$ to the observed space y by noising the function value $y = f_{\boldsymbol{\theta}}(\boldsymbol{x}) + \epsilon$. However, frequentists assume that there are some true parameter values $\boldsymbol{\theta}^*$, which can be obtained by optimizing an objective $L_{\boldsymbol{\theta}}(y, \hat{y})$, where $\hat{y}$ are the predictions. E.g. minimizing some kind of loss

$$\boldsymbol{\theta}_{\text{opt}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_n L_{\boldsymbol{\theta}}(y_n, \hat{y}_n) \, . \tag{1.2.1}$$

The hope is that the model $f_{\boldsymbol{\theta}_{\text{opt}}}$ resembles the true model with parameters $\boldsymbol{\theta}^*$, which is hidden from us. The Bayesians on the other hand define a likelihood, which is similar to the objective of the frequentists (however, more generalized: the objective can be constructed by some statistics of the likelihood[5]). But instead of optimization, Bayesians assume a prior distribution $p(\boldsymbol{\theta})$ on the parameters $\boldsymbol{\theta}$ and infer the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ of the parameters conditioned on the observed data $\mathcal{D}$ by Bayes rule

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \, . \tag{1.2.2}$$

That way they do not approximate the true parameters $\boldsymbol{\theta}^*$ by a single point $\boldsymbol{\theta}_{\text{opt}}$, which summarizes the data, but summarize the data with a full distribution $p(\boldsymbol{\theta}|\mathcal{D})$. By not modeling the data with a single point estimate, but by many (a distribution), Bayesians naturally incorporate uncertainties that arise due to the limited data observed so far. By observing more and more data, the posterior will collapse in the limit of infinite data to a delta function at the true parameter $\boldsymbol{\theta}^*$ (as long as the prior has a finite support for $\boldsymbol{\theta}^*$). If true values are not supported by the prior then the posterior converges to a delta function that supports values that minimize the KL-divergence between true data density and the inferred data density. This effect is referred to as asymptotic certainty.

In addition to the ease of incorporation of uncertainties, Bayesian framework also has a unique way of model averaging. Where as the frequentist framework suggests one best (according to some measure) function that fits the data, Bayesians average over different functions, weighted by the explainability and complexity of the function. This leads to an automated Occams razor effect (MacKay, 2002; Rasmussen and Ghahramani, 2001). This point is further discussed in later chapters. In this thesis, we choose the Bayesian way, which might be better suited if uncertainties are to be quantified. In their paper Bayarri and Berger (2004) discuss further the interplay of Bayesian and frequentist methods.

[5] The maximum/mode of the likelihood, for example, can be an objective.

1.3 Structure of the thesis

The first part of the thesis (Chapter 2 to Chapter 6) introduces the tools used in the work during the PhD. Some of the work in this thesis has been published in Nirwan and Bertschinger (2019a,b). This thesis provides a more extended version of the published papers. The extensions are included in Chapter 7, Chapter 8 and Chapter 10. Chapter 9 is again an introduction to the tools needed for Chapter 10. Finally, Chapter 11 summarizes and concludes the work in this thesis.

We start with the basics of Bayesian modeling in Chapter 2 and continue building upon that. In Chapter 3 methods for approximating the posterior are presented. Here, we also look into the tools available for approximate inference. Chapters 4 and 5 give a short introduction to Gaussian processes and Gaussian process latent variable models. Chapter 6 introduces financial concepts and models that we use in the proceeding chapter. In Chapter 7 we describe and extend the work on the usage of Gaussian process latent variable models for financial modeling, which is published in Nirwan and Bertschinger (2019b). Chapter 8 is a description and extension of the work published in Nirwan and Bertschinger (2019a), which deals with a problem arising due to the rotation invariancy in latent space models. The tools for the final work on quantile matching estimation (order statistics) are introduced in Chapter 9. Chapter 10 builds upon that and introduces a Bayesian method for matching a distribution to observed quantile values (Nirwan and Bertschinger, 2020).

The code for the experiments in this thesis is available at https://github.com/rsnirwan.

Chapter 2

Bayesian modeling

As motivated in the Introduction (Chapter 1), probabilities provide a very nice way to deal with uncertainties. In this chapter, we look into the fundamentals of Bayesian modeling by first introducing Bayes rule and the predictive distribution and then applying the learning to the simplest model: Linear regression. We fit the linear regression model first within the frequentist framework and then within the Bayesian framework. The goal is to understand the Bayesian philosophy and how it is connected to the frequentist point of view.

2.1 Introduction

Let's first start with the description of a model. In a more general sense a model is an approximation of a system. When we are dealing with data, the model describes a set of data one could observe from the modeled system. It defines a data generating process. Learning happens, when we allow many different models to be true in principle and the data is choosing the best of them based on some kind of goodness measure (frequentist) or weights them by their ability to generate a particular kind of data (Bayesian)[1].

So one of the biggest differences between frequentist and Bayesian is that the former chooses one particular model for prediction, but the latter takes all of the models from a particular model class. The prediction of all the models is weighted based on the ability of the models to generate the observed data. So we assume that a given data set $\mathcal{D}$ is generated by a model m. Since we do not know, how the model m looks like that generated $\mathcal{D}$, we have to reverse engineer starting from the data $\mathcal{D}$ and ending with the right model m. This process is called inference. We infer the right model m based on the data $\mathcal{D}$ we saw. This, however, is not a one-to-one mapping. Because of the scarcity of the data and the noise in the data, there might be more than just one model m that generated the data set $\mathcal{D}$. So, we can never be sure about the true model m and have somehow to incorporate that uncertainty.

Bayesian modeling provides a way to deal with that uncertainty. Instead of taking just one model m, we allow for more than one model to be true. This is done by assigning a

[1] As we will see, this weighting is performed with respect to the posterior distribution.

probability (score) $p(m)$ to each model m based on our prior believes for the ability of m to generate similar data sets to $\mathcal{D}$. Doing so incorporates the uncertainty related to the scarcity of the data. The uncertainty related to the noise in the data can be captured by assigning a probability $p(\mathcal{D}|m)$ to the data set $\mathcal{D}$ generated by a particular model m[2]. The reverse engineering that leads to models with high probability of generating the observed data $\mathcal{D}$ is done using Bayes rule

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)p(m)}{p(\mathcal{D})} \; . \tag{2.1.1}$$

This quantity is called the posterior. It is the updated believe (updated prior $p(m)$) for the models that could have generated data sets like $\mathcal{D}$. The update is performed by the likelihood $p(\mathcal{D}|m)$. So, the prior is changed through the likelihood to the posterior. The quantity in the denominator $p(\mathcal{D})$ is called the marginal likelihood. It is a constant with respect to the model m and renormalizes the product of the likelihood and the prior.

At this point every information about the data is contained in the posterior. The posterior is our updated belief about which models are more likely than others to have generated $\mathcal{D}$. For prediction we use all the models that the posterior allows and weight them by their score (posterior probability) that the posterior puts on each of them. The probability of new data $\mathcal{D}^*$ conditioned on the already seen data $\mathcal{D}$ is given by

$$p(\mathcal{D}^*|\mathcal{D}) = \int p(\mathcal{D}^*|m)p(m|\mathcal{D})\mathrm{d}m \; . \tag{2.1.2}$$

These two equations, the posterior in Equation (2.1.1) and the predictive distribution in Equation (2.1.2), build the foundation of Bayesian modeling.

This chapter introduces the basics of Bayesian modeling. First, we start with the frequentist version of linear regression. After that we extend that to the Bayesian version.

2.2 Linear regression

To understand probabilistic modeling or Bayesian machine learning we will start with the simplest model: Linear Regression. Linear Regression is easy to understand and to interpret, but its flexibility is very limited. Nevertheless, it is very helpful to understand the connection and differences between frequentists and Bayesians. In the chapter about Gaussian processes, we will also build upon the linear regression model.

In this subsection we solve the Linear Regression problem first with the frequentists approach and then within the Bayesian framework and compare both approaches. We provide a brief review here. For a more detailed description, see Bishop (2006).

In supervised machine learning settings, we are given a data set $\mathcal{D}$ of N input points $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T$, where each $\boldsymbol{x}_n \in \mathbb{R}^D$ and N corresponding targets $\boldsymbol{y} = (y_1, \ldots, y_N)^T$.

[2] Not just the prior $p(m)$, but also the likelihood $p(\mathcal{D}|m)$ makes prior assumptions about the data generating process. The prior allows or discards models by placing a finite or zero probability on models. The likelihood on the other hand makes assumption about the noise around a particular model. This is an assumption about the variability of the data $\mathcal{D}$ that is not captured by the model m.

In a regression task y_n is a continous ($y_n \in \mathbb{R}$) and in a classification task y_n is discrete (in binary classification for example $y_n \in \{0, 1\}$).

2.2.1 Frequentists linear regression

In the frequentists setting, one tries to model the data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ by fitting a function $f_{\boldsymbol{w}}$ parameterized by $\boldsymbol{w}$ [3]. The targets $\boldsymbol{y}$ are assumed to be noisy. The modeling assumption is that the observed values are coming from a true data generating function f_{true} and are corrupted by a noise. The goal is to find/learn a function that is close to the true data generating function. In other words: Take the set of parameters $\boldsymbol{w}$ that results in a function closest to all of the data points (points we have seen so far but also that we have not seen yet). So, we want to find the function that generalises best.

Model

In the case of linear regression, we fit a linear function to the data $\mathcal{D}$ of the form

$$y_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n) + \epsilon_n = \boldsymbol{w}^T \boldsymbol{x}_n + \epsilon_n, \tag{2.2.1}$$

where ϵ_n is the noise. This type of model is very limited in its flexibility, since it only can pick up linear relationship between the input $\boldsymbol{x}_n$ and the target y_n. Therefore, one extends the model by a linear combination of fixed nonlinear functions $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_D)^T$ of the input. Equation (2.2.1) then becomes

$$y_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n) + \epsilon_n = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) + \epsilon_n \ . \tag{2.2.2}$$

As already mentioned, the goal is to find the best approximation within the model class to the true function f_{true}. The information we have about f_{true} is coming only from the observed data $\mathcal{D}$. Thus, we have to define a metric that uses our modeling choice $f_{\boldsymbol{w}}$ and the observed data $\mathcal{D}$ and encodes our understanding of what “best” means.

The idea is the following: We want the predictions $\hat{y}_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n)$ of our model $f_{\boldsymbol{w}}$ to be close to the observed data, since they are coming from f_{true}. So, we define a function L that takes $\hat{y}_n$ and y_n as inputs and outputs a value related to their closeness. This function is the penalty for the prediction $\hat{y}_n$ being different from the observed value y_n and is called a loss function or error function. One of the most used loss functions is the mean-squared error (MSE)

$$L_{n,\text{MSE}} = L(y_n, \hat{y}_n) = (y_n - \hat{y}_n)^2 \ . \tag{2.2.3}$$

We can now define the total loss as the mean of the losses over all observed points. However, only minimizing the loss defined that way leads to overfitting if the model $f_{\boldsymbol{w}}$ is too flexible. Therefore, we have to penalize too flexible functions by including a regularization term to

[3] One can also use one of the many non-parametric models (e.g. Support Vector Machines, Random Forest, ...). In this section, however, we will only focus on the parametric linear regression.

the total loss. The regularization term is responsible to keep the weights $\boldsymbol{w}$ low and thus prohibit unnecessary flexibility. For a $l2$ regularization the total loss becomes

$$\begin{aligned} L_{\mathcal{D}}(\boldsymbol{w}) &= \frac{1}{N}\sum_{n=1}^{N} L_n + \lambda\|\boldsymbol{w}\|_2^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}(y_n - f_{\boldsymbol{w}}(\boldsymbol{x}_n))^2 + \lambda\|\boldsymbol{w}\|_2^2, \end{aligned} \tag{2.2.4}$$

where $\lambda \in \mathbb{R}^+$ is a constant and determines the tradeoff between bias and variance. There is a direct correspondence between this loss function and the Bayesian linear regression[4]. This point is further discussed in Chapter 2.2.2.

Learning

Learning means to find the optimal parameters $\boldsymbol{w}_{\text{opt}}$ for the data set $\mathcal{D}$ and the model $f_{\boldsymbol{w}}$ under the given metric (e.g. MSE with $l2$ regularization). In particular

$$\boldsymbol{w}_{\text{opt}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, L_{\mathcal{D}}(\boldsymbol{w}) \ . \tag{2.2.5}$$

In the case of linear regression (where the model can be written as in Equation (2.2.2)) there is a closed form solution to the minimisation problem (Bishop, 2006)

$$\boldsymbol{w}_{\text{opt}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} - \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y} \ , \tag{2.2.6}$$

where for M basis functions $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \dots, \phi_M(\boldsymbol{x}))^T$ the matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$ is defined as

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & & \phi_M(\boldsymbol{x}_1) \\ & \ddots & \\ \phi_1(\boldsymbol{x}_N) & & \phi_M(\boldsymbol{x}_N) \end{pmatrix} . \tag{2.2.7}$$

For a more complicated/flexible model, when there is no analytical expression for the optimal weights $\boldsymbol{w}_{\text{opt}}$, one has to elude to iterative gradient methods. Ruder (2016) provides a nice overview of many gradient descent optimization algorithms.

Left plot in Figure 2.2.1 shows a toy data set created with a third order polynomial as the true function. Figure 2.2.2 shows how the MSE changes when we change the intercept of a linear function without any nonlinear basis functions according to the gradient of the loss-function. The MSE is getting lower the closer we get to the data points. By choosing nonlinear basis functions in addition and minimizing the error according to the weights of all the basis functions, we can even further decrease the MSE. This is shown in the right hand side of Figure 2.2.1, where the red line represents the fit with polynomial basis functions up to order three.

[4] The MSE-function corresponds to a Gaussian noise model and the $l2$ regularization corresponds to a Gaussian prior for the weights in Bayesian linear regression.

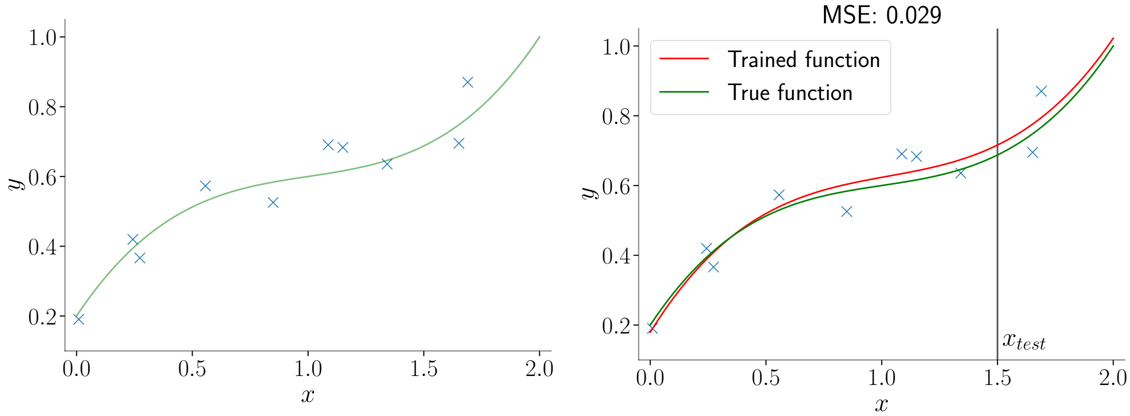


Figure 2.2.1: Left: Toy data set with a third order polynomial as the true funtion. Right: linear regression fit with polynomial basis function up to order three.

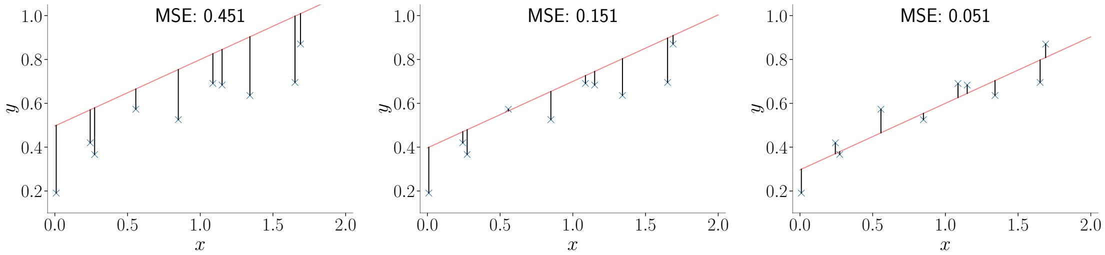


Figure 2.2.2: Gradient decent applied to the intercept of a linear model.

Prediction

After we have learned the optimal parameters $\boldsymbol{w}_{\text{opt}}$, we can make predictions for input we have not seen so far. For that, we evaluate the function $f_{\boldsymbol{w}_{\text{opt}}}$ with the optimal parameters $\boldsymbol{w}_{\text{opt}}$ at the new input location $\boldsymbol{x}_{\text{test}}$

$$y_{\text{pred}} = f_{\boldsymbol{w}_{\text{opt}}}(\boldsymbol{x}_{\text{test}}) \ . \tag{2.2.8}$$

The right side of Figure 2.2.1 shows the input location with the vertical back line. The predicted value is the cut of the black line with the trained function (red). The true value is the cut with the true function (green).

There is an error when we take the value of the red line as the prediction, which is the distance of the red line to the true function (green). However, this error cannot be quantified in this setting. First of all because in general we do not know the true function (green) but also because we are not modeling the uncertainties that arise due to the scarcity of the data and the observation noise. These points are addressed by the Bayesian framework, that we look into in the next section.

2.2.2 Bayesian linear regression

Also in the Bayeisan setting, the goal is to model the observed data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$. However, in contrast to the frequentists approach, Bayesians use probabilities for modeling. We already discussed the advantages of using probabilities for modeling in the beginning of this chapter. One of which is the ability to average over all possibilities of the model class. Whereas the frequentists take one out of all possible models from the model class $f_{\boldsymbol{w}}$, Bayesian take all of them and weight the prediction for each of them based on some metric. This metric is the posterior distribution for each set of the parameters $\boldsymbol{w}$. All this naturally arises from only the sum and product rule of probabilities.

The sum rule states that the marginal distribution is given by the summation (for discrete) or the integral (for continuous) of the other variable

$$p(x) = \int p(x, y) \mathrm{d}y \; . \tag{2.2.9}$$

The second one is the product rule

$$p(x, y) = p(y|x)p(x) \; , \tag{2.2.10}$$

which states that the joint distribution can be decomposed into a conditional and a marginal distribution. Those two basic rules will allow us to learn the parameter distribution after the observation of the data and also allow us to make predictions for unseen data. Let us first start by defining the model and we will come back to the sum and product rule in the learning/inference and the prediction section.

Model

As already mentioned, the goal of Bayesian machine learning is also to model the observed data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$. Therefore, also here we need to make an assumptions for the form of the function we believe the data is coming form. For the linear model, that assumption is the same as in the frequentists case (Equation 2.2.1).

$$y_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n) + \epsilon_n = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) + \epsilon_n \; . \tag{2.2.11}$$

where $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_D)^T$ are fixed nonlinear functions of the input and ϵ_n is the noise[5]. But now, instead of defining a penalty for the deviation from $f_{\boldsymbol{w}}$ as the MSE in the frequentists case, we define a probability function that tells us, how probable it is to see a point in a certain distant from the function $f_{\boldsymbol{w}}$. Bayesian linear regression chooses a Gaussian distribution with mean $f_{\boldsymbol{w}}$. The standard deviation σ of this Gaussian distribution

[5] Note how we separated the variability in the data into noise and structure. Equation (2.2.11) defines the noise as the variability in the data which cannot be captured by $f_{\boldsymbol{w}}$. If we would change $f_{\boldsymbol{w}}$ to some other function $\tilde{f}_{\boldsymbol{w}}$ it would pick up different variability in the data and thus the variability denoted as noise would change as well. So, without the model specifications (which also includes the likelihood and the prior), there is no interpretation of the data. That makes it even more important to know your model, as all the information we get from our data is subjective to the definition of the model. First footnote of Chapter 5 includes some more thoughts similar to this one.

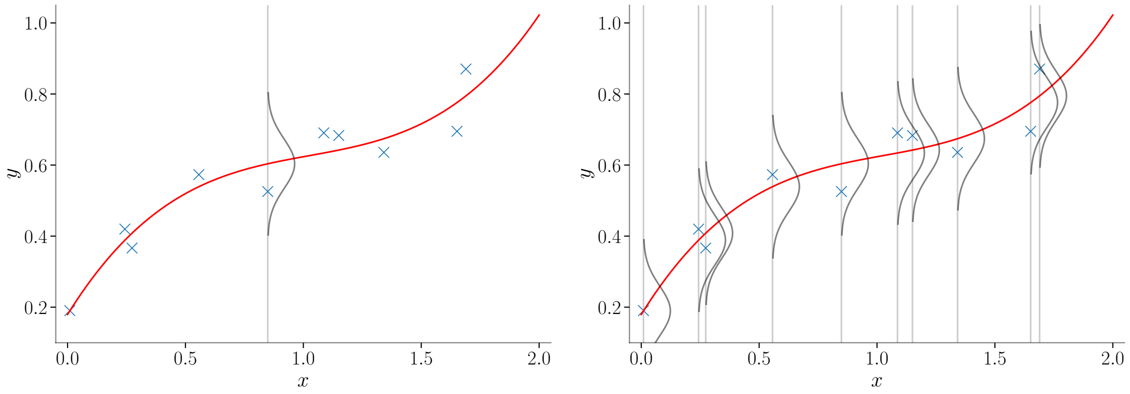


Figure 2.2.3: Illustration of the likelihood function.

determines, how much data points can depart from $f_{\boldsymbol{w}}$. So, for a single data point $\boldsymbol{x}_n$ this distribution takes the from

$$p(y_n|\boldsymbol{w},\sigma) = \mathcal{N}\left(y_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n),\sigma\right) \ . \tag{2.2.12}$$

This equation is called the likelihood for a single data point $(\boldsymbol{x}_n, y_n)$ and is illustrated for a single data point in Figure 2.2.3. The black Gaussian in the plot is bound to the red function and its width is given by σ. It defines, how likely it is to observe the data point for a particular choice of $\boldsymbol{w}$ and σ.

The likelihood for all of the N data points, if we make the assumption that the data is drawn for the same model independently (i.i.d. - independent and identically distributed), is then given by the product of the likelihood of each single point[6]

$$p(\boldsymbol{y}|\boldsymbol{w},\sigma) = \prod_{n=1}^{N} p(y_n|\boldsymbol{w},\sigma) = \prod_{n=1}^{N} \mathcal{N}\left(y_n|\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n),\sigma\right) \ . \tag{2.2.13}$$

Right plot in Figure 2.2.3 illustrates the likelihood for all data points. The Gaussian "bumps" kind of behave like a tunnel. The width of the tunnel (departure from the mean $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)$) is given by σ. If for a set of parameters $\boldsymbol{w}$ the "tunnel" is too far away from the data points, the model assigns small likelihood value to that model. On the other hand if the likelihood value is high for a set $\boldsymbol{w}$, the set $\boldsymbol{w}$ gets a higher likelihood score. Thus, it will also contribute more to the expected value under the posterior[7].

The maximum likelihood solution $\boldsymbol{w}_{\text{opt}} = \text{argmax}_{\boldsymbol{w}}\, p(\boldsymbol{y}|\boldsymbol{w},\sigma)$ leads to the MSE[8]. Here we see the correspondence between the two methods: Linear regression with MSE assumes a

[6] Note that we do not assume the data to be i.i.d. in general but the data to be i.i.d. **conditioned** on the input and the parameters. This leads to an i.i.d. assumption only for the noise, not the data. If the data were i.i.d. learning would not be possible. Also, the model assumption that the conditioning lead to i.i.d. data means that, all the information we want to get out of the data has to be encapsulated in the parameters.

[7] The likelihood score is scaled by the prior score that we assign to a particular $\boldsymbol{w}$ before we take the expectation.

[8] $\text{argmax}_{\boldsymbol{w}}\, p(\boldsymbol{y}|\boldsymbol{w},\sigma) = \text{argmin}_{\boldsymbol{w}} \sum_{n=1}^{N}\left(y_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2$

Gaussian noise around the learned function $f_{\boldsymbol{w}}$. As a Bayesian, however, learning is not associated with optimization of objective functions like the likelihood. Instead, learning is the use of Bayes rule. For that a crucial part of the model is still missing. Before we can use Bayes rule, we need to specify a prior for the unknowns (parameters) of our model. The prior encodes our beliefs about the possible values of the parameters before we see the data. In the linear regression case the prior for the parameters $\boldsymbol{w}$ is also a Gaussian

$$p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \alpha \boldsymbol{I}) \ , \tag{2.2.14}$$

where $\alpha \in \mathbb{R}^+$ determines the magnitude of the parameters $\boldsymbol{w}$. It acts a regularizer and is equivalent to λ in Equation (2.2.4), penalizing high value of $\boldsymbol{w}$. So, with the likelihood (Equation 2.2.13) and the prior (Equation 2.2.14) we have defined the full model. The next step is learning.

Learning/Inference

Bayes rule (Equation 2.1.1) tells us how to make inference over the unknown parameters from the data

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y})} \ , \tag{2.2.15}$$

where in the numerator on the right hand side we have the likelihood of the data $p(\boldsymbol{y}|\boldsymbol{w})$ and the prior $p(\boldsymbol{w})$. The denominator $p(\boldsymbol{y})$ is called the marginal likelihood and is given by the sum rule $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})\mathrm{d}\boldsymbol{w}$. The term $p(\boldsymbol{w}|\boldsymbol{y})$ on the left hand side is the posterior of our parameters. The posterior is our updated belief of the parameters after seeing the data $\boldsymbol{y}$ [9]. The Gaussian prior is a so called conjugate prior to the Gaussian likelihood in the linear model for fixed σ. For conjugate priors the posterior also has the same functional form as the prior and is analytically tractable. In our case the Gaussian posterior of the parameters $\boldsymbol{w}$ is given by (Bishop, 2006)

$$p(\boldsymbol{w}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N) \ , \tag{2.2.16}$$

where $\boldsymbol{m}_N$ and $\boldsymbol{S}_N$ are the mean and the covariance of our parameters $\boldsymbol{w}$ after seeing the N data points and are given by

$$\begin{aligned} \boldsymbol{m}_N &= \frac{1}{\sigma^2}\boldsymbol{S}_N \boldsymbol{\Phi}^T \boldsymbol{y} \\ \boldsymbol{S}_N &= \left(\alpha^{-1}\boldsymbol{I} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1} \ , \end{aligned} \tag{2.2.17}$$

where $\boldsymbol{\Phi}$ is defined in Equation (2.2.7). Note that for a Gaussian distribution the mean and the mode are the same point. The mode of the posterior is also called the maximum a posteriori (MAP). In this particular example the MAP estimate is given by the mean $\boldsymbol{m}_N$. The mode of the likelihood function only is called the maximum likelihood estimate.

[9] We start with the prior as our belief of the parameters without seeing the data and the likelihood changes our beliefs. The updated belief is then given by the posterior.

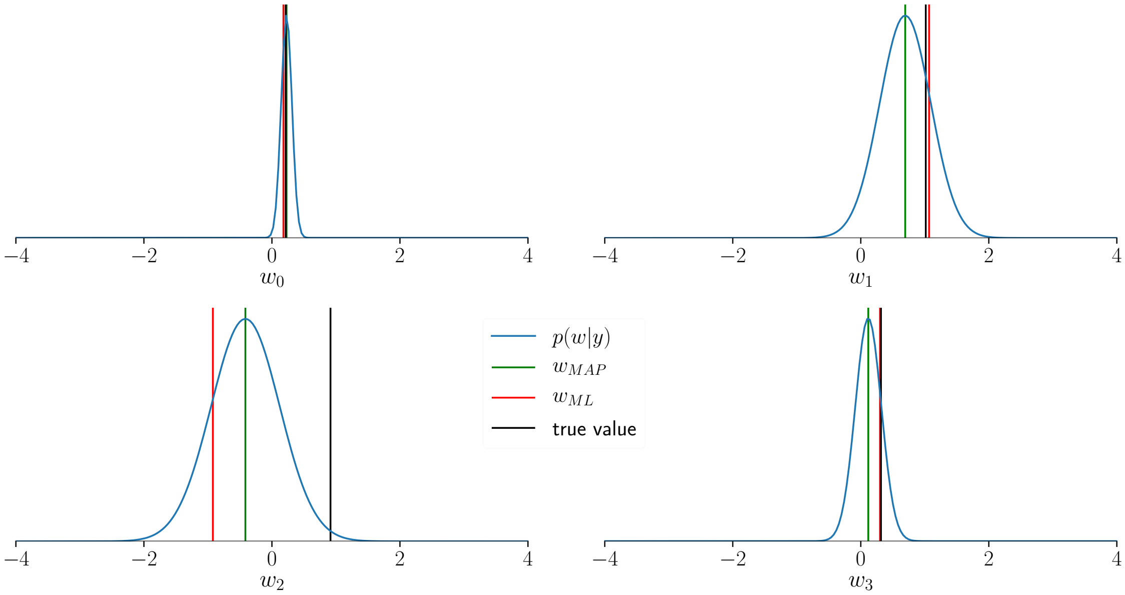


Figure 2.2.4: The posterior $p(\boldsymbol{w}|\boldsymbol{y})$ is show in blue. Maximum likelihood estimate of the parameters is shown with the red vertical line, the MAP estimate is the green line and the true parameters are shown with the black vertical line.

The maximum likelihood estimate of the linear regression yields the same solution as the optimized solution in the frequentists setting (Equation 2.2.6) without regularization. The optimized solution with regularization (finite λ) corresponds to the MAP estimate[10].

Now we fit the same data as in the previous section (Figure 2.2.1) again. The marginal posterior distribution $p(w|\boldsymbol{y})$ for each w of a third order polynomial fit is shown in Figure 2.2.4 together with the true value (black line) and the MAP and maximum likelihood (ML) estimate (green and red line). In contrast to the other estimates, the fully Bayesian treatment provides the full posterior distribution including the uncertainty over the weights $\boldsymbol{w}$. This quantification of the uncertainty over $\boldsymbol{w}$ can now be used to not only make prediction for unseen data but also to quantify the uncertainty in the prediction.

Prediction

After inferring the posterior the prediction together with the prediction uncertainty (predictive distribution) $p(y_{\text{pred}}|\boldsymbol{y})$ for a test data point $\boldsymbol{x}_{\text{test}}$ is given by the integral of the

[10] For corresponding α, λ and σ:
$\boldsymbol{m}_N = \operatorname{argmax}_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{y}) = \operatorname{argmin}_{\boldsymbol{w}} \left[\frac{1}{N}\sum_{n=1}^{N}(y_n - f_{\boldsymbol{w}}(\boldsymbol{x}_n))^2 + \lambda\|\boldsymbol{w}\|_2^2\right]$

likelihood of the test point with respect the posterior distribution[11]

$$p(y_{\text{pred}}|\boldsymbol{y}) = \int p(y_{\text{pred}}|\boldsymbol{w})\ p(\boldsymbol{w}|\boldsymbol{y})\mathrm{d}\boldsymbol{w}\ . \tag{2.2.18}$$

This is also what is meant by averaging over all possible models. The integration is weighting all parameters $\boldsymbol{w}$ for the prediction, which are allowed by the prior $p(\boldsymbol{w})$ and the likelihood $p(\boldsymbol{y}|\boldsymbol{w})$. Also the ones that under the frequentists setting would have considered as really bad (huge loss). Those "bad" fits are included by the Bayeisan framework but their posterior weight (contribution to the predictive distribution) will be very low. On the other hand, functions that fit the data perfectly (have very low loose according to e.g. MSE) but are very complex/flexible are also not weighted high. In those cases the likelihood score might be high but the prior will weight their contribution down. The idea that simple explanations are usually better than complicated ones is called Occam's Razor and is naturally embodied in Bayesian statistics (Jefferys and Berger, 1992; Rasmussen and Ghahramani, 2001).

Since in the linear case everything is Gaussian with respect to $\boldsymbol{w}$, we can also calculate the predictive distribution for an input $\boldsymbol{x}_{\text{test}}$ in a closed form (Bishop, 2006)

$$p(y_{\text{pred}}|\boldsymbol{x}_{\text{test}}, \boldsymbol{y}) = \mathcal{N}\left(y_{\text{pred}}|\boldsymbol{m}_N^T \boldsymbol{\phi}(\boldsymbol{x}_{\text{test}}), \sigma_N^2(\boldsymbol{x}_{\text{test}})\right)\ , \tag{2.2.19}$$

where

$$\sigma_N^2(\boldsymbol{x}_{\text{test}}) = \sigma^2 + \boldsymbol{\phi}(\boldsymbol{x}_{\text{test}})^T \boldsymbol{S}_N \boldsymbol{\phi}(\boldsymbol{x}_{\text{test}}) \tag{2.2.20}$$

is the variance of the predictive distribution. Left hand side of Figure 2.2.5 shows the mean and two standard deviations of the predictive distribution. Samples from the predictive distribution are shown on the right hand side.

So in theory the Bayesian modeling approach is really easy. After defining the model (likelihood and the prior) we do inference using Bayes rule and get predictions by averaging the likelihood for the test data point under the posterior. However the posterior and the predictive distribution are rarely analytically tractable due to the intractable high dimensional integrals for the marginal likelihood $p(\boldsymbol{y})$[12]. Since there is no analytical solution in most of the cases, we have to use approximation methods for these integrals. Those approximation methods are discussed in Chapter 3.

2.2.3 Summary

The goal of machine learning is to is to learn a function $f_{\boldsymbol{w}}$, parametrized by $\boldsymbol{w}$, that takes an input $\boldsymbol{x}_n$ and outputs a prediction $\hat{y}_n$. Learning basically means to constantly adjust the parameters $\boldsymbol{w}$ in a certain way until we are "satisfied" with the output of the predictor.

[11] Note the following equality that comes from the product rule $p(y_{\text{pred}}, \boldsymbol{w}|\boldsymbol{y}) = p(y_{\text{pred}}|\boldsymbol{w}, \boldsymbol{y})p(\boldsymbol{w}|\boldsymbol{y})$. Thus $p(y_{\text{pred}}|\boldsymbol{y}) = \int p(y_{\text{pred}}|\boldsymbol{w}, \boldsymbol{y})\ p(\boldsymbol{w}|\boldsymbol{y})\mathrm{d}\boldsymbol{w}$. However, the assumption is made, that all the information about $\boldsymbol{y}_{\text{pred}}$ is contained in the parameters $\boldsymbol{w}$ and therefore we can write $p(y_{\text{pred}}|\boldsymbol{w}, \boldsymbol{y}) = p(y_{\text{pred}}|\boldsymbol{w})$.

[12] In models where the prior and the posterior are from the same family of distribution (conjugate models) the posterior is tractable. We will have a closer look into one of these models called the Gaussian process in Chapter 4.

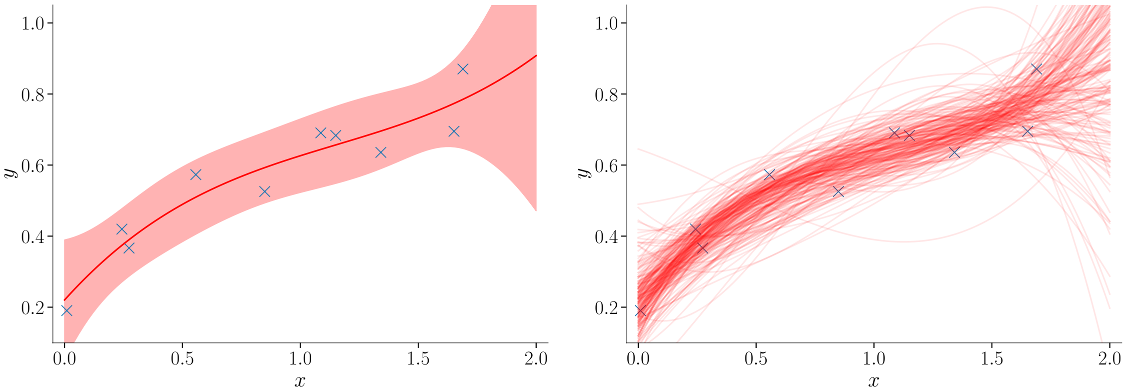


Figure 2.2.5: Left: Predictive distribution $p(\boldsymbol{y}^*|\boldsymbol{y})$. Right: 200 Samples from the predictive distribution.

To measure the "satisfaction" we define a loss-function $L_n = L(y_n, \hat{y}_n)$. L is defined in such a way, that the closer the predicted value $\hat{y}_n$ is to the observed value y_n the smaller is the loss L_n. The total loss is then given by the mean of all losses and a regularization term $L_{\mathcal{D}}(\boldsymbol{w}) = \frac{1}{N}\sum_n L_n + \lambda\|\boldsymbol{w}\|_2^2$. The optimal parameters $\boldsymbol{w}_{\text{opt}}$ of the predictor are the ones for which the total loss $L_{\mathcal{D}}(\boldsymbol{w})$ is minimized. Prediction for an unseen test point $\boldsymbol{x}_{\text{test}}$ is then given by evaluating the function $f_{\boldsymbol{w}_{\text{opt}}}$ at the test point $f_{\boldsymbol{w}_{\text{opt}}}(\boldsymbol{x}_{\text{test}})$.
In the Bayesian case, the model is given in terms of probabilities, which defines, how likely it is to observe the data given specific parameters $p(\boldsymbol{y}|\boldsymbol{w})$ and a belief about the parameters before seeing the data $p(\boldsymbol{w})$. Learning is simply given by the use of Bayes rule and results into the updated belief about the parameters after seeing the data $p(\boldsymbol{w}|\boldsymbol{y})$. Prediction for an unseen test point $\boldsymbol{x}_{\text{test}}$ is given by the expectation of the likelihood for the test point under the updated beliefs (posterior distribution).
A direct comparison of the frequentist and the Bayesian approach is given in the table below.

Comparison of frequentist and Bayesian framework

Frequentist Framework

- Data

$$\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$$
$$\boldsymbol{x}_n \in \mathbb{R}^D,\ y \in \mathbb{R}$$

- Model

$$y_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n) + \epsilon_n = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) + \epsilon_n$$

$$\begin{aligned} L_{\mathcal{D}}(\boldsymbol{w}) =& \frac{1}{N}\sum_{n=1}^{N}(y_n - f_{\boldsymbol{w}}(\boldsymbol{x}_n))^2 \\ &+ \lambda \|\boldsymbol{w}\|_2^2 \end{aligned}$$

- Fit

$$\boldsymbol{w}_{\text{opt}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, L_{\mathcal{D}}(\boldsymbol{w})$$

- Prediction

$$y_{\text{pred}} = f_{\boldsymbol{w}_{\text{opt}}}(\boldsymbol{x}_{\text{test}})$$

Bayesian Framework

- Data

$$\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$$
$$\boldsymbol{x}_n \in \mathbb{R}^D,\ y \in \mathbb{R}$$

- Model

$$y_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n) + \epsilon_n = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) + \epsilon_n$$

$$p(\boldsymbol{y}|\boldsymbol{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n | f_{\boldsymbol{w}}(\boldsymbol{x}_n), \sigma^2)$$
$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \boldsymbol{I})$$

- Inference

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{w})\ p(\boldsymbol{w})}{p(\boldsymbol{y})}$$

- Prediction

$$p(y_{\text{pred}}|\boldsymbol{y}) = \int p(y_{\text{pred}}|\boldsymbol{w})\ p(\boldsymbol{w}|\boldsymbol{y}) \mathrm{d}\boldsymbol{w}$$

Chapter 3

Approximations to the posterior

The fully Bayesian treatment of the model, where we infer the posterior $p(\boldsymbol{w}|\boldsymbol{y})$ of the parameters $\boldsymbol{w}$ and integrate them out to get the predictive distribution $p(y_{\text{pred}}|\boldsymbol{y})$ is preferable but not always possible. Therefore, we have to resort to approximation methods. This chapter is a short review on Markov chain Monte Carlo (MCMC) methods and Variational Bayes (VB). In particular we review the Hamiltonian Monte Carlo (HMC) algorithm for sampling and Variational inference (VI) for approximating the posterior by a variational distribution.

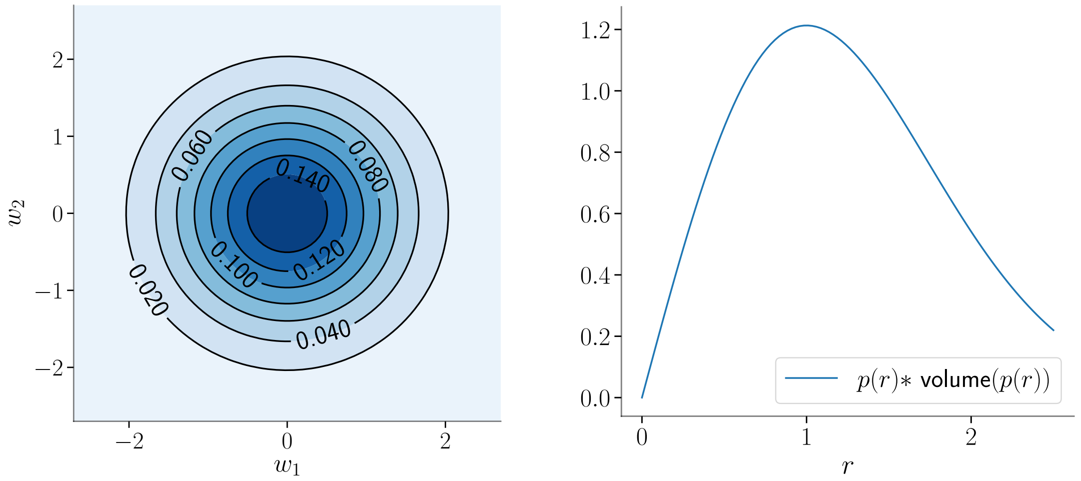
The posterior distribution is the object that captures all the information contained in our data. The whole Bayesian procedure comes down to calculate expectations of functions with respect to the posterior distribution

$$\mathbb{E}[f] = \int f(\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y})\mathrm{d}\boldsymbol{w} \ . \tag{3.0.1}$$

The model specification is not hard, but its solution is, i.e. the integration/expectation calculation with respect to the posterior. Because these expectations are not analytically tractable, we resort to approximation methods.

The main idea behind those approximation methods is to only focus on the parts that contribute largely to the integral. Most of the space does actually not contribute to the integral. Either the posterior value or the volume of the space corresponding to the posterior value will be too small. This is illustrated in Figure 3.0.1 with a 2d standard Gaussian. Left side of Figure 3.0.1 shows the conture lines with constant probability density. Not the area with the highest probability density is important but the area, where the product of the probability density with the area of the corresponding density is important and contribute most to the integral. The product of both is shown on the right hand side. Here we see, that the maximum of the product is not where the maximum of the probability density lies. In Higher dimensions even less volume is concentrated around high values of the probability density[1]. So actually we can neglect the space where the probability mass is low and only concentrate on the regime, that most contribute to the integral (the typical set).

[1] Note that we distinguish between the posterior density $p(\boldsymbol{w}|\boldsymbol{y})$ and the posterior mass $p(\boldsymbol{w}|\boldsymbol{y})\mathrm{d}\boldsymbol{w}$.



(a) Posterior density of a standard 2d Gaussian.

(b) Posterior mass for different level sets.

Figure 3.0.1: Illustration of the maximum of the probability density vs. the level sets, that contribute most to the integral. For the standard Gaussian case the different radii correspond to different level sets.

The approximation algorithms for the posterior inference can be divided into two big classes. The first one approximates the integral for the predictive distribution using samples from the posterior. Those methods are called sampling methods. The second big class approximates the posterior with another distribution which is similar to the posterior but easier to handle. This technique is known as variational approximation.
In this chapter we will look into sampling and variational Bayes. In particular, we will look closer into how Hamiltonian Monte Carlo (a sampling technique) and how variational inference work.

3.1 Sampling

3.1.1 Monte Carlo sampling

One way to find the typical set are Monte Carlo methods, where we draw samples according to the mass of the typical set. In one dimensional case, the mode has the most probability mass. However, this is not true anymore for higher dimensions[2].
The popularity of Monte Carlo methods is due to the fact that the samples that we take from the posterior density are already points that correspond to the typical set with high probability mass. After having those samples, we can approximate the integral/expecta-

[2] A Gaussian distribution for example has only for one dimension the most mass at the mode. For higher dimensions, the typical set the away from the mode and the distance between the mode and the typical set is proportional to $\sqrt{D-1}$, where D is the dimension of the Gaussian. Figure 3.0.1 illustrates this for a 2d Gaussian.

tion of a function f by the evaluation of the function at those samples

$$\mathbb{E}[f] = \int_{\mathbb{R}^D} f(\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y})\mathrm{d}w \simeq \int_T f(\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y})\mathrm{d}w \simeq \frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{w}_i) \,, \tag{3.1.1}$$

where T is the typical set (with high probability mass) and M the number of samples drawn from the posterior $p(\boldsymbol{w}|\boldsymbol{y})$. The estimate of the expectation becomes more accurate the longer the chain gets. The quality of Monte Carlo method can be evaluated in term of its bias and variance. The estimation by Monte Carlo methods is unbiased, meaning $\mathbb{E}[\hat{\mu}] = \mathbb{E}[f]$, where $\hat{\mu}$ is the estimated value and $\mathbb{E}[f]$ is the true value. Also the variance can be quantified. The estimation of the expectation follows a Gaussian with a variance that decreases with the total number of the samples M

$$\frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{w}_i) \sim \mathcal{N}\left(\mathbb{E}[f], \frac{\mathrm{var}[f]}{M}\right) \,. \tag{3.1.2}$$

This is a huge advantage and makes Monte Carlo integration effective in high dimensions.

Inversion Sampling: The simplest sampling method is the inversion sampling. In inversion sampling samples u from a uniform distribution are transformed to samples from a desired distribution p by applying the inverse of the cumulative distribution function P of p on the samples. $P^{-1}(u)$ will have the desired distribution.

Rejection Sampling: Another method to sample from a distribution without the need for the cumulative distribution is rejection sampling. Here the target distribution p is enclosed by another density cq, where q is a probability density and c a constant such that $cq(x) > p(x)\ \forall\ x$. The samples are taken from the distribution q but for each x only the proportion $p(x)/(cq(x))$ is kept, the rest is rejected. High values of c are needed to fulfill the condition $cq(x) > p(x)\ \forall\ x$, but the proportion of rejected samples is also increasing with c. A huge advantage of rejection sampling is that we do not even need p to be normalized. It also works with unnormalized densities. However, it has the inefficiency coming from the rejected samples. This inefficiency is fixed by importance sampling.

Importance Sampling: Importance sampling also samples from q to approximate the integral with respect to p. But instead of rejecting some of the samples to get the desired distribution, importance sampling assigns weights to each of the samples. The weight for each sample x is given by $p(x)/q(x)$. The disadvantage of this method is, that it fails if p has fatter tails that q. In this case relatively few samples are taken from the tails and those samples might have very high weights since $p(x)/q(x)$ can become arbitrarily large.

To use the above listed methods one either needs the inverse of the cumulative distribution or a guess for q that is close to the target distribution. If neither is given they cannot be used or become very inefficient. There is, however, another class of methods that allow us to sample from a target distribution even if the two conditions are not satisfied. Those are called Markov chain Monte Carlo (MCMC) methods.

3.1.2 Markov chain Monte Carlo (MCMC)

In contrast to the above mentioned methods, MCMC methods produce a chain of dependent samples instead of independent samples. However, the stationary distribution of the chain still follows the desired distribution. A sequence of random variables $x_1, x_2, \dots$ is called a Markov chain, if the successor x_{i+1} of x_i is independent of $\{x_j\}_{j=1}^{i-1}$ conditioned on x_i $\forall$ i, i.e: $p(x_{i+1}|x_1, \dots, x_i) = p(x_{i+1}|x_i)$. A MCMC method yields a stationary distribution, if the following conditions are fulfilled:
Detailed balance: The detailed balance property requires the transition probability p_{ij} from state x_i to x_j and the transition probability p_{ji} from state x_j to x_i to fulfill the following equation: $p(x_i)\ p_{ij} = p(x_j)\ p_{ji}$. A Markov process satisfying detailed balance is also called a reversible Markov process.
Ergodicity: A Markov chain is ergodic if there is always a positive probability to pass from any state to any other state.
Mixing: A slightly stronger condition than the Ergodicity is the mixing, which requires the convergence to the same stationary distribution independent of the starting point. So, whatever the starting point is, the chain always converges to the same distribution.
There are many MCMC approaches. E.g.: Gibbs sampling, that reduces sampling from a multidimensional distribution to a sequence of 1d or Metropolis-Hasting, that draws a new sample x (next element of the chain) by using a proposal distribution $p(x|x')$ that is conditioned on the previous sample x'. Besag et al. (1995) discuss the basic methodology of MCMC including algorithms like Gibbs sampling and Metropolis-Hasting. In our work we approximate the posterior using the probabilistic programming language Stan (introduced in Section 3.3), that uses a version of Hamiltonian Monte Carlo (HMC). Therefore, our focus in this section will be on HMC.

3.1.3 Hamiltonian Monte Carlo

In this section we will briefly review Hamiltonian Monte Carlo (HMC). It has been described in detail in Betancourt (2017). HMC is constructing chains that are moving toward the typical set and exploring its mass. If we want to sample $\boldsymbol{w} \in \mathbb{R}^D$ from the desired distribution $p(\boldsymbol{w}|\boldsymbol{y})$ (i.e. the posterior distribution that we are interested in) using HMC, we introduce other D auxiliary variables $\boldsymbol{\nu} = \frac{\partial \boldsymbol{w}}{\partial t}$ to the set of our parameters $\boldsymbol{w} \in \mathbb{R}^D$. Those auxiliary variables are called the moments of $\boldsymbol{w}$ and the $\boldsymbol{w}$-$\boldsymbol{\nu}$-space is called the phase space. Also the target distribution $p(\boldsymbol{w}|\boldsymbol{y})$ is “lifted up” to a higher dimension

$$\begin{aligned} \boldsymbol{w} &\to (\boldsymbol{w}, \boldsymbol{\nu}) \\ p(\boldsymbol{w}|\boldsymbol{y}) &\to p(\boldsymbol{w}, \boldsymbol{\nu}|\boldsymbol{y}) \ . \end{aligned} \tag{3.1.3}$$

After using the product rule of probabilities on the joint distribution $p(\boldsymbol{w}, \boldsymbol{\nu}|\boldsymbol{y}) = p(\boldsymbol{w}|\boldsymbol{\nu}, \boldsymbol{y})p(\boldsymbol{\nu}|\boldsymbol{y})$ we can map it to a physical system by defining

$$\begin{aligned} H(\boldsymbol{w}, \boldsymbol{\nu}) &= -\log\left(p(\boldsymbol{\nu}|\boldsymbol{w}, \boldsymbol{y})p(\boldsymbol{w}|\boldsymbol{y})\right) \\ &= \underbrace{-\log p(\boldsymbol{\nu}|\boldsymbol{w}, \boldsymbol{y})}_{K} \underbrace{-\log p(\boldsymbol{w}|\boldsymbol{y})}_{V} \ , \end{aligned} \tag{3.1.4}$$

where K is the kinetic energy and V is the potential energy. The potential energy $V = -\log p(\boldsymbol{w}|\boldsymbol{y})$ is given by our Bayesian model but the kinetic energy K is not. Different choices for K yield different algorithms for HMC. We choose K to be a quadratic function of $\boldsymbol{\nu}$, i.e. $K = \frac{\boldsymbol{\nu}^2}{2m}$. This is analogous to the momentum of a particle with mass m and yields a Gaussian density for $p(\boldsymbol{\nu}|\boldsymbol{w}, \boldsymbol{y})$. By this choice the $\boldsymbol{\nu}$ is independent of $\boldsymbol{w}$ and $\boldsymbol{y}$ and $p(\boldsymbol{\nu}|\boldsymbol{w}, \boldsymbol{y}) = p(\boldsymbol{\nu})$. Therefore, if we marginalize out the momentum $\boldsymbol{\nu}$ we immediately recover our target distribution $\int p(\boldsymbol{w}, \boldsymbol{\nu}|\boldsymbol{y})\mathrm{d}\boldsymbol{\nu} = p(\boldsymbol{w}|\boldsymbol{y})$. Moreover, this ensures that the projection onto $\boldsymbol{w}$ of the trajectories exploring the typical set of the phase space distribution also explore the typical set of the target distribution $p(\boldsymbol{w}|\boldsymbol{y})$. The Hamiltonian 3.1.4 captures the geometry of the typical set and Hamilton's equations of motion provide a way to sample the typical set (Betancourt, 2017). After having specified the Hamiltonian $H = K + V = \text{const}$, we can use Hamilton's equations of motion to create a trajectory

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \frac{\partial H}{\partial \boldsymbol{\nu}}$$
$$\frac{\mathrm{d}\boldsymbol{\nu}}{\mathrm{d}t} = -\frac{\partial H}{\partial \boldsymbol{w}} \,. \tag{3.1.5}$$

First we choose some $\boldsymbol{w}$ at random and repeat the following as many times as the number of samples we want from our posterior:

1. sample $\boldsymbol{\nu} \sim \mathcal{N}(0, 1)$
2. Solve for the trajectory with $H(\boldsymbol{w}, \boldsymbol{\nu})$ for some amount of time[3]
3. Save $\boldsymbol{w}$ as a sample, where $\boldsymbol{w}$ is the end point of the trajectory[4]

The chain of $\boldsymbol{w}$'s will converge to the posterior $p(\boldsymbol{w}|\boldsymbol{y})$. This is how we sample p using HMC in theory. In practice, however, we integrate (step 2) the trajectory numerically and numerical integrators tend to drift away from the exact solution because of the finite step size. To preserve detailed balance, despite of the instability, we have to correct this error. The error is corrected if we only accept the proposal with probability $\min(1, \exp\left(H(\boldsymbol{w}_{\text{end}}, -\boldsymbol{\nu}_{\text{end}}) - H(\boldsymbol{w}_{\text{start}}, \boldsymbol{\nu}_{\text{start}})\right)$ (Betancourt, 2017). If the new proposal is rejected, the next sample is again the current sample and is counted again when estimating the expectations of a function (Neal, 2010). The ergodicity condition is also fulfilled by this algorithm. Since the momentum $\boldsymbol{\nu}$ is sampled from a Gaussian (see step 1) and can take on any value, the position $\boldsymbol{w}$ can be affected in arbitrary ways.

[3] This amount of time is called the integration time. If it is to short, we might need a lot of time to explore the full space. On the other hand if it is too large, the trajectory might end up in already previously explored neighborhoods. Stan implements the No-U-Turn termination criterion. Using this criterion, the integration continues as long as the trajectory is expanding towards phase space regions away from where it comes from. As soon as the momentum at beginning and the end of trajectory are anti-aligned (the trajectory moves backwards) the criterion is satisfied and the trajectory stops expanding (Hoffman and Gelman, 2014).

[4] The points of the trajectory live in the phase space $(\boldsymbol{w}, \boldsymbol{\nu})$. The marginalization of $\boldsymbol{\nu}$ is equivalent to ignoring the $\boldsymbol{\nu}$ part of the trajectory. Therefore, we only need to save $\boldsymbol{w}$.

HMC, in comparison to Metropolis-Hasting and Gibbs sampling, reduces significantly the correlation between successive samples. Because of the reduction in correlation we need fewer samples to approximate the integral with the same accuracy as, for example, in Metropolis Hasting and Gibbs sampling. Other huge advantages of sampling methods in general are, that they are unbiased and have a convergence guarantee. Moreover, they are easy to implement with the toolboxes available today. HMC also does not require the target density to be normalized. It is enough to know the density up to the normalization constant. In our case we want to sample from the posterior, which is not tractable due to the infeasible marginal likelihood that normalizes the product of the likelihood and the prior. Fortunately, we do not need it when using HMC.

However, the drawbacks of HMC are that, even though we have a convergence guarantee, the convergence to the target distribution can take a long time. Especially if the distribution is multimodal, samplers can get stuck in only one of the modes. Solutions to the problem of exploring multimodal distributions are suggested but they also further increase the computational complexity of the problem. Betancourt (2014) suggests to transform the complex posterior to a simpler distribution using a measure preserving deformation utilizing the geometry of equilibrium thermodynamic processes and Hoffman et al. (2019) suggest also to transform the complex posterior to a simpler distribution using normalizing flows (Papamakarios et al., 2019). In both cases one can then build a Markov chain of samples on the simpler distribution and then transform the samples according to the inverse transformation. When using very flexible transformations, the multimodal posterior can be transformed to an unimodal distribution, where sampling becomes really easy. However, the complexity does not disappear with these methods but is just shifted towards finding flexible, yet easy to invert transformations with simple Jacobians.

Another class of methods that are faster than MCMC, but do not have the convergence guarantees anymore are the class of Variational methods that we will look into in the next section.

3.2 Variational inference

The idea of Variational inference is also to approximate the posterior distribution. This is not done with samples but with another distribution within a family of distributions, called the variational distribution. There are many variational methods on the market (Ambrogioni et al., 2018; Li and Turner, 2016; Minka, 2001). In this section, however, we will focus on a particular method where the "distance" between the variational distribution and the true posterior is minimized using the Kullback-Leibler divergence as the loss (Bishop, 2006; MacKay, 2002). Zhang et al. (2019) review recent trends in the field of VI.

For that we define a variational distribution $q_{\boldsymbol{\nu}}(\boldsymbol{w})$, where $\boldsymbol{\nu}$ are the parameters of the variational distribution, called the variational parameters. The task is to find the set of parameters such that the $q_{\boldsymbol{\nu}}(\boldsymbol{w})$ is close to the true posterior $p(\boldsymbol{w}|\boldsymbol{y})$ as much as possible.

The closeness of the distributions is measured in the Kullback-Leibler divergence[5]

$$\mathrm{KL}(q||p) = \int q_{\boldsymbol{\nu}}(\boldsymbol{w}) \log \frac{q_{\boldsymbol{\nu}}(\boldsymbol{w})}{p(\boldsymbol{w}|\boldsymbol{y})} \mathrm{d}\boldsymbol{w} \ . \tag{3.2.1}$$

The benefits of using variational inference compared to MCMC methods are that variational inference is deterministic, converges faster and needs far less iterations than MCMC. On the other hand there are no convergence guarantees like the reduction of the variance of the expectation with more number of samples.

Derivation of variational inference

For the derivation of variational inference we need Jensen's inequality for concave functions. It states that the average of the function f applied at points $\boldsymbol{x}$ is smaller than the function f applied to the average of $\boldsymbol{x}$

$$f(\mathbb{E}[\boldsymbol{x}]) \geq \mathbb{E}[f(\boldsymbol{x})] \ . \tag{3.2.2}$$

The application of Jensen's inequality to the data density $p(\boldsymbol{y})$ (also called the marginal likelihood or the evidence) leads to the KL objective function

$$\begin{aligned}
\log p(\boldsymbol{y}) &= \log \int p(\boldsymbol{y}, \boldsymbol{w}) \mathrm{d}\boldsymbol{w} \\
&= \log \int p(\boldsymbol{y}, \boldsymbol{w}) \frac{q_{\boldsymbol{\nu}}(\boldsymbol{w})}{q_{\boldsymbol{\nu}}(\boldsymbol{w})} \mathrm{d}\boldsymbol{w} \\
&= \log \mathbb{E}_{q_{\boldsymbol{\nu}}(\boldsymbol{w})} \left[\frac{p(\boldsymbol{y}, \boldsymbol{w})}{q_{\boldsymbol{\nu}}(\boldsymbol{w})} \right] \\
&\geq \mathbb{E}_{q_{\boldsymbol{\nu}}(\boldsymbol{w})} \left[\log \frac{p(\boldsymbol{y}, \boldsymbol{w})}{q_{\boldsymbol{\nu}}(\boldsymbol{w})} \right] \\
&= \int q_{\boldsymbol{\nu}}(\boldsymbol{w}) \log \frac{p(\boldsymbol{y}, \boldsymbol{w})}{q_{\boldsymbol{\nu}}(\boldsymbol{w})} \mathrm{d}\boldsymbol{w} \\
&:= \mathrm{ELBO} \ .
\end{aligned} \tag{3.2.3}$$

The quantity at the end of the expression is a lower bound on the marginal likelihood, called the ELBO (Evidence lower bound). It turns out that the difference between the marginal likelihood $p(\boldsymbol{y})$ and the ELBO is exactly the Kullback-Leibler divergence from

[5] Although the Kullback-Leiber divergence is seen as a way to measure distance between probability distributions, it is not a true metric. It is not symmetric, i.e. $\mathrm{KL}(q||p) \neq \mathrm{KL}(p||q)$.

the variational family $q_{\boldsymbol{\nu}}(\boldsymbol{w})$ to the true posterior $p(\boldsymbol{w}|\boldsymbol{y})$

$$\begin{aligned}\mathrm{KL}(q_{\boldsymbol{\nu}}(\boldsymbol{w})\|p(\boldsymbol{w}|\boldsymbol{y})) &= \int q_{\boldsymbol{\nu}}(\boldsymbol{w})\log\frac{q_{\boldsymbol{\nu}}(\boldsymbol{w})}{p(\boldsymbol{w}|\boldsymbol{y})}\mathrm{d}\boldsymbol{w}\\ &= \int q_{\boldsymbol{\nu}}(\boldsymbol{w})\log\frac{q_{\boldsymbol{\nu}}(\boldsymbol{w})p(\boldsymbol{y})}{p(\boldsymbol{w},\boldsymbol{y})}\mathrm{d}\boldsymbol{w}\\ &= \int q_{\boldsymbol{\nu}}(\boldsymbol{w})\log\frac{q_{\boldsymbol{\nu}}(\boldsymbol{w})}{p(\boldsymbol{w},\boldsymbol{y})}\mathrm{d}\boldsymbol{w} - \int q_{\boldsymbol{\nu}}(\boldsymbol{w})\log p(\boldsymbol{y})\mathrm{d}\boldsymbol{w}\\ &= -\mathrm{ELBO} + p(\boldsymbol{y})\ .\end{aligned} \tag{3.2.4}$$

Here we see again that the ELBO is a lower bound to the marginal likelihood. Since the KL divergence on the left hand side is always non-negative, we get $p(\boldsymbol{y}) \geq \mathrm{ELBO}$.
Recall, that we want to approximate $p(\boldsymbol{w}|\boldsymbol{y})$ with $q_{\boldsymbol{\nu}}(\boldsymbol{w})$ by minimizing the KL divergence. And now we found another method to do that. Since the marginal likelihood $p(\boldsymbol{y})$ is a constant with respect to $\boldsymbol{w}$, instead of minimizing the KL divergence we maximize the ELBO. The procedure is to first define the likelihood $p(\boldsymbol{w}|\boldsymbol{y})$ and the prior $p(\boldsymbol{w})$, then choose a family of distributions $q_{\boldsymbol{\nu}}(\boldsymbol{w})$ and at the end maximize the ELBO with respect to the variational parameters $\boldsymbol{\nu}$. Note that the higher the ELBO the closer $q_{\boldsymbol{\nu}}(\boldsymbol{w})$ is to the true posterior. The ELBO takes its maximum value $p(\boldsymbol{y})$ when $q_{\boldsymbol{\nu}}(\boldsymbol{w}) = p(\boldsymbol{w}|\boldsymbol{y})$. In that case we are performing exact inference. Thus, we can not overfit the data by choosing a very flexible family as the variational distribution[6]. Thus we can choose any variational family to increase the ELBO as much as possible. After maximizing the ELBO we use the variational approximation $q_{\boldsymbol{\nu}}(\boldsymbol{w})$ (instead of the posterior $p(\boldsymbol{w}|\boldsymbol{y})$) for further analysis of the model, e.g. for predictions. The $\mathrm{KL}(q_{\boldsymbol{\nu}}(\boldsymbol{w})\|p(\boldsymbol{w}|\boldsymbol{y})$ has three important cases, we need to understand to get an intuition of what is happening when optimizing with respect to $\boldsymbol{\nu}$. If q is high and p is high the cost will be low. If q is high but p is low, then the cost is high (leads to an decrease of q in that area, when performing optimization). If q is low, because of the expectation over q, the cost is low independent of the value of p. The last point leads to an approximation that is called mode splitting. A variational distribution with e.g. only one mode will try to capture just one mode of the posterior as good as possible and set $q \simeq 0$ on the other modes without increasing the cost. So, by minimizing $\mathrm{KL}(q\|p)$ we will not match the whole probability density p but only the part with the most mass.
One of the famous family of variational distributions is the Gaussian distribution, where the multivariate posterior distribution $p(\boldsymbol{w}|\boldsymbol{y})$ is fitted by independent Gaussians

$$q_{\boldsymbol{\nu}}(\boldsymbol{w}) = \prod_{d=1}^{D} q_{\boldsymbol{\nu}}(\boldsymbol{w}_i) = \prod_{d=1}^{D}\mathcal{N}(\boldsymbol{w}_d|\mu_d,\sigma_d)\ . \tag{3.2.5}$$

It has a nice analogy to physics: the solution to the mean field approximation of the ising model can exactly be reconstructed by the independent Gaussians as the variational family (MacKay, 2002).

[6] This is not the same as in the frequentists setting, where choosing a very flexible model and maximizing the objective function (without proper regularization) can lead to overfitting. Here, we are variationally protected against overfitting.

Kucukelbir et al. (2017) introduces a framework called automatic differentiation variational inference (ADVI), that can be used as a black-box for variational inference with Gaussian variational distribution (either mean-field approximation of a multivariate Gaussian with a full covariance matrix). The idea is to first unconstrain all latent space variables (unbound distributions with bounded support with a fixed transformation) and then fit a Gaussian in the unconstrained space. ADVI is also used in the probabilistic programming language Stan that we use for our experiments.

3.3 Probabilistic programming

Fortunately enough, we do not need to implement these approximations by ourself to get started with Bayesian machine learning. Software tools such as probabilistic programming languages (e.g. Stan (Carpenter et al., 2017)) or other libraries for automatic differentiation (e.g. Tensorflow (Abadi et al., 2015), Pytorch (Paszke et al., 2019)) significantly simplify the implementation of the models discussed earlier and allow us to use approximation methods out of the box.

Particularly Stan makes the implementation of the Bayesian models very easy. Stan is a framework for Bayesian inference, that tries to abstract away the heavy math behind the approximation methods (HMC and VI) that we described earlier, such that we can only focus on the modeling part. We only need to specify the posterior density and Stan internally uses automatic differentiation for the gradients needed for HMC and VI. It also provides a strong diagnostic at the end of the inference procedure when using HMC.

Below are some examples (linear model and the Gaussian process model that we discuss in the next chapter). We just need to define the right blocks. Data-block contains the input from the outside (the knowns). Parameter-block contains the parameters that we want to sample (the unknowns) and the model-block defines the relation between the knowns and unknowns via the likelihood and the priors. If there is a need to define some intermediate variables, which depend on the parameters and are needed for the model block, they can be defined in the transformed-parameter-block.

In the examples below we see the statistical model defined on the left hand side and the corresponding code on the right hand side. Note the similarity between the Bayesian model on the left side with the model block in the stan code. Also the code for the Gaussian process regression is included, which is introduced in Chapter 4. Here, we make use of the transformed-parameter-block to calculate the kernel matrix $\boldsymbol{K}$[7].

3.4 Summary

In this chapter we discussed two variants of approximating the posterior. The first methods approximates the posterior by samples from the posterior. Here, we looked at Hamiltonian Monte Carlo (HMC), that uses Hamilton's equation of motion to build a correlated

[7] Instead of working directly with the covariance matrix $\boldsymbol{K}$, one would actually work with the Cholesky decomposition of $\boldsymbol{K}$, which is numerically more stable. Here, we skip this step, because the code is for illustration purposes only.

Bayesian linear regression

$$\begin{aligned} \boldsymbol{w}_d &\sim \mathcal{N}(0,1)\ \forall\ d \\ \sigma &\sim \mathcal{N}(0,1) \\ y_n|\boldsymbol{w},\sigma^2 &\sim \mathcal{N}(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n),\sigma^2)\ \forall\ n \end{aligned}$$

```
data{
    int<lower=0> N;
    int<lower=0> D;
    matrix[N, D] X;
    vector[N] y;
}
parameters{
    vector[D] w;
    real<lower=0> sigma;
}
model{
    w ~ normal(0,1);
    sigma ~ normal(0,1);
    y ~ normal(X*w, sigma);
}

```

Gaussian process regression

$$\begin{aligned} \sigma &\sim \mathcal{N}(0,1) \\ l &\sim \mathcal{N}(0,1) \\ \sigma_{\text{noise}} &\sim \mathcal{N}(0,1) \\ \boldsymbol{y}|\sigma,l,\sigma_{\text{noise}} &\sim \mathcal{N}(\boldsymbol{0},\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma_{\text{noise}}\boldsymbol{I}) \end{aligned}$$

```
data{
    int<lower=0> N;
    int<lower=0> D;
    vector[N] y;
    vector[D] X[N];
}
parameters{
    real<lower=0> sigma;
    real<lower=0> l;
    real<lower=0> sigma_noise;
}
transformed parameters{
    matrix[N, N] K = cov_exp_quad
   (X, sigma, l);
    for (i in 1:N)
        K[i, i] += square(
   sigma_noise);
}
model{
    sigma ~ normal(0,1);
    l ~ normal(0,1);
    sigma_noise ~ normal(0,1);
    y ~ multi_normal(rep_vector
   (0, N), K);
}

```

sequence of samples that has the desired target distribution. Instead of the integration, we average the function values evaluated at the sampled values that we obtain from HMC. This leads to an unbiased estimate of our expectation. The more samples we take the accurate our expectation will be. Moreover, the density, we are sampling from, does not need to be normalized. This means that we can sample from the posterior, without calculating the marginal likelihood. On the other hand, HMC is very slow compared to other methods, e.g. variational inference.

Variational inference (VI) was the second approximation method that we looked into. VI, in contrast to HMC, approximate the actual distribution not by samples but by another simpler distribution. We start by suggesting a family of distributions and from all the distributions in the family we choose the one that minimizes its distance to the target distribution. The distance measure we looked into was the Kullback leibler divergence. The simpler distribution can then be taken to calculate the expectations/integral. The biggest advantages of using variational inference are that it is deterministic, it converges quite fast and requires very few iterations compared to MCMC methods. However, variational inference does not have any convergence guarantees as MCMC methods.

At the end, we saw, how easy it is to implement probabilistic models using the tools available to us. We provided code for Bayesian linear regression and the Gaussian process regression in the probabilistic programming language Stan. We only have to define the likelihood and the prior and the approximations (both HMC and VI) can be used out of the box, without much knowledge about the heavy math behind the approximation methods.

Chapter 4

Gaussian processes

Gaussian Processes are non-parametric kernel based approch to regression. We already discussed approximate inference methods in Chapter 3. We use them because the integrals we need to solve for Bayesian inference are most of the time not tractable. Gaussian processes, however, have the huge advantage that the first level inference is analytically tractable. This is due to the nice properties of the Gaussian distribution.

In this chapter we start with the Gaussian distribution and then go to the infinite dimensional Gaussian distribution, the Gaussian process. As for the Gaussian distribution, also for Gaussian processes the covariance between different dimensions plays an important role. We discuss the role and try to give intuition on how the covariance influences the regression function. Thereafter, we analyse some particular forms of the covariance structure (stationary covariance functions). Those can be decomposed into their Fourier components, which lead to a better understanding of the model. We end with a brief outlook into some advanced topics like the sparse Gaussian processes for computational speedup.

4.1 Gaussian process as a limit to Gaussian distribution

Gaussian processes (GPs) are the limit of a multidimensional Gaussian distributions when we increase the dimensionality of the multivariate Gaussian to infinity. Thus, to understand GPs we need to understand Gaussian distributions first. The probability density of a N dimensional multivariate Gaussian distribution is given by

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N|\boldsymbol{\Sigma}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right\}, \tag{4.1.1}$$

where $\boldsymbol{\mu}$ is the mean of the Gaussian and $\boldsymbol{\Sigma}$ is the covariance matrix. If the diagonal (the variance of each dimension) only consists of ones, the matrix is also called the correlation matrix. The off-diagonal elements of the correlation matrix basically tell us, how much the value of a point in two different dimensions correlate with each other. The close the value is to one, the higher the correlation. Higher correlation means that once we know one of the values, we also have information about the value of the other dimension.

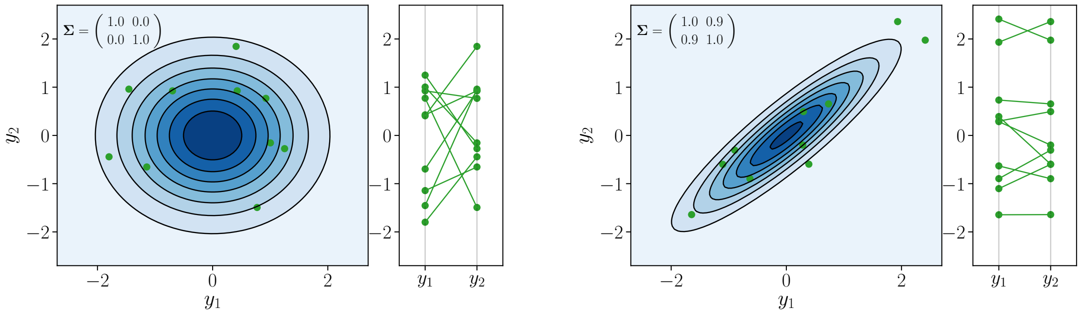


Figure 4.1.1: Two dimensional Gaussian distribution.

Figure 4.1.1 shows two 2-dimensional multivariate Gaussian distributions. The off-diagonal elements of the Gaussian on the left hand side are zero. That means, for a sample from this distribution, if we know one element, we have no information about the other element at all. On the right hand side, we see a Gaussian with higher covariance. Here, if we know one element of a sample, we have information about the other element as well. The positive correlation means that, if one element is positive, then with high probability the other element will be positive too.

The Gaussian distribution with dimensionality larger than two cannot be visualized in this way anymore. Therefore, we need another representation. For our purposes, it is enough to visualize samples from the distribution. The connected green points on the right hand side of each subfigure of Figure 4.1.1 show samples drawn from the corresponding Gaussian. Their corresponding draw is also shown in green in the 2-d density plot. The vertical black lines in the background each represent a dimension of the Gaussian. For 2-dimensional Gaussian, we have two lines and each sample has one position (the value of the sample for that dimension) on each line. In the case, where the correlation is zero, the elements of the same sample are arbitrarily distributed across the lines. But the interesting case is, where the correlation is high (right hand side of Figure 4.1.1). In this case the elements of each sample are near to each other. That is highlighted by the connection of the elements, which on the right hand side are almost always horizontal compared to the left hand side. Left hand side of Figure 4.1.2 shows three samples from an 11 dimensional Gaussian. Different colors indicate different samples. The covariance in this case is chosen in a way, that reflects the distance between the points. Points near to each other have a higher correlation than points further away. In particular

$$\boldsymbol{y} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} k_{11} & \cdots & k_{1N} \\ \vdots & \ddots & \vdots \\ k_{N1} & \cdots & k_{NN} \end{pmatrix}\right), \tag{4.1.2}$$

where we choose $k_{ij} = k(x_i, x_j) = \exp\left\{-\frac{1}{2}(x_i - x_j)^2\right\}$. $k(\cdot,\cdot)$ is called the kernel function. This particular form is called radial basis kernel function (RBF) or the squared exponential kernel. If we further increase the dimension of the multivariate Gaussian, we end up with an infinite dimensional Gaussian called a Gaussian process (GP). The samples from a GP

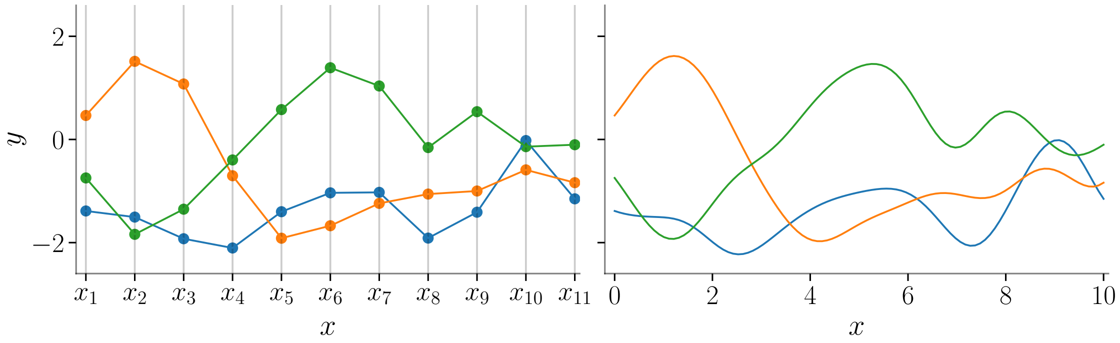


Figure 4.1.2: Left: Three samples from an 11 dimensional multivariate Gaussian. Right: Three sampled functions from a Gaussian process.

are functions. Three samples are shown on the right hand side of Figure 4.1.2. We denote the distribution as

$$\boldsymbol{y} \sim \mathcal{GP}(m(x), k(x, x')) \ , \tag{4.1.3}$$

where $m(x)$ is the mean function of the GP and $k(x, x')$ is the covariance/kernel function.

Definition 4.1.1 *A Gaussian process is a collection of random variables $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N)$, if they have a joint Gaussian distribution for any finite selection $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$.*

Like a Gaussian distribution, which is completely specified by its mean vector and the covariance matrix, a Gaussian process is also completely specified by its mean function and covariance function. In GP regression, the mean function is often set to zero[1]. The kernel function contains information about how the GP evaluated at x and x' covary. The properties of the sampled function (i.e. smoothness, periodicity) are determined by the choice of the kernel function. We will look into different kernel functions later in this chapter.

4.2 Gaussian process regression

In last section we gave a visual introduction to Gaussian processes. In this section we will look into a bit more mathy introduction. We start again with the Bayesian linear regression model from Chapter 2.2.2. Its generalization will lead to Gaussian processes. The view through the linear regression model, where the weights to the basis functions are an important quantity, is called the weight space view. The Gaussian processes, however, do not contain weighted basis functions, which compose the actual function, but directly work in the function space. This view is called the function space view. We will briefly review both views. Detailed description is provided by (Rasmussen and Williams, 2005).

[1] The posterior Gaussian process mean, as we will see, can still have a non-zero mean.

4.2.1 Weight space view

In Chapter 2.2.2 we started with the basis function approach to linear regression

$$y_n = f_{\boldsymbol{w}}(\boldsymbol{x}_n) + \epsilon_n = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}_n) + \epsilon_n \; . \tag{4.2.1}$$

The likelihood of the model is given by an independent Gaussian noise assumption $p(\boldsymbol{y}|\boldsymbol{w}) = \prod_n \mathcal{N}(y_n|f_{\boldsymbol{w}}(\boldsymbol{x}_n), \sigma)$ around the function $f_{\boldsymbol{w}}$ and the prior in $\boldsymbol{w}$ is also a Gaussian. The posterior $p(\boldsymbol{w}|\boldsymbol{y})$ and the predictive distribution $p(y_{\text{pred}}|\boldsymbol{y})$ for a test point $\boldsymbol{\phi}_* = \boldsymbol{\phi}(\boldsymbol{x}_{\text{test}})$ are both analytically tractable in the Gaussian case and are given by

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{y}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}_N, \boldsymbol{S}_N) \\ p(y_{\text{pred}}|\boldsymbol{y}) &= \mathcal{N}\left(y_{\text{pred}}|\boldsymbol{m}_N^T \boldsymbol{\phi}_*, \sigma_N^2(\boldsymbol{x}_{\text{test}})\right) \; , \end{aligned} \tag{4.2.2}$$

where

$$\begin{aligned} \boldsymbol{m}_N &= \frac{1}{\sigma^2} \boldsymbol{S}_N \boldsymbol{\Phi}^T \boldsymbol{y} \\ \boldsymbol{S}_N &= \left(\alpha^{-1} \boldsymbol{I} + \frac{1}{\sigma^2} \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \\ \sigma_N^2(\boldsymbol{x}_{\text{test}}) &= \sigma^2 + \boldsymbol{\phi}_*^T \boldsymbol{S}_N \boldsymbol{\phi}_* \; . \end{aligned} \tag{4.2.3}$$

Note that σ^2 in $\sigma_N^2(\boldsymbol{x}_{\text{test}})$ adds the independent noise on top of the predicted function f' (i.e. $y_{\text{pred}} = f' + \epsilon$). The predictive function distribution without the noise is

$$p(f'|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{m}_N^T \boldsymbol{\phi}_*, \boldsymbol{\phi}_*^T \boldsymbol{S}_N \boldsymbol{\phi}_*\right) \; . \tag{4.2.4}$$

If we look closer into the mean and covariance of $p(f'|\boldsymbol{y})$ we get (Rasmussen and Williams, 2005)[2]

$$\begin{aligned} \boldsymbol{m}_N^T \boldsymbol{\phi}_* &= \alpha \boldsymbol{\phi}_*^T \boldsymbol{\Phi}(\sigma^2 \boldsymbol{I} + \alpha \boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{y} \\ \boldsymbol{\phi}_*^T \boldsymbol{S}_N \boldsymbol{\phi}_* &= \alpha \boldsymbol{\phi}_*^T \boldsymbol{\phi}_* - \alpha \boldsymbol{\phi}_*^T \boldsymbol{\Phi}(\alpha \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \sigma^2 \boldsymbol{I})^{-1} \alpha \boldsymbol{\Phi}^T \boldsymbol{\phi}_* \; . \end{aligned} \tag{4.2.5}$$

Here we see, that the predictive distribution is only depending on the inner product of the evaluated basis functions at $\boldsymbol{x}_{\text{test}}$ and $\boldsymbol{x}$. Since only the inner product of the basis functions $\boldsymbol{\phi}$ evaluated at different points $\boldsymbol{x}$ and $\boldsymbol{x}'$ is necessary, we can directly define a function for the inner product $k(\boldsymbol{x}, \boldsymbol{x}')$. This is known as the kernel trick, which lifts the input space $\boldsymbol{x}$ up to a feature space by replacing inner products of basis functions $\boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{x}')$ with kernel functions $k(\boldsymbol{x}, \boldsymbol{x}')$.
By replacing $\alpha \boldsymbol{\phi}_*^T \boldsymbol{\phi}_*$ with $\boldsymbol{K}_{**}$, $\alpha \boldsymbol{\phi}_*^T \boldsymbol{\Phi}$ with $\boldsymbol{K}_*$, $\alpha \boldsymbol{\Phi}^T \boldsymbol{\Phi}$ with $\boldsymbol{K}$ and $\alpha \boldsymbol{\Phi}^T \boldsymbol{\phi}_*$ with $\boldsymbol{K}_*^T$ in Equation (4.2.5), and allowing for multiple predictions $\boldsymbol{f}' = (f_{*1}, \dots, f_{*N'}) \in \mathbb{R}^{N'}$, we get

$$p(\boldsymbol{f}'|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{K}_*(\sigma^2 \boldsymbol{I} + \boldsymbol{K})^{-1} \boldsymbol{y}, \boldsymbol{K}_{**} - \boldsymbol{K}_*(\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{K}_*^T\right) \; . \tag{4.2.6}$$

We started with the basis functions with weights and arrived at an expression where we marginalized out the weights analytically and found another form of expressing the basis functions through kernels. Next chapter briefly outlines another derivation, which does not use basis functions at all.

[2] For the covariance matrix we use the matrix inversion lemma
$(\boldsymbol{Z} + \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T)^{-1} = \boldsymbol{Z}^{-1} - \boldsymbol{Z}^{-1}\boldsymbol{U}(\boldsymbol{W}^{-1} + \boldsymbol{V}^T \boldsymbol{Z}^{-1} \boldsymbol{U})^{-1} \boldsymbol{V}^T \boldsymbol{Z}^{-1}$.

4.2.2 Function space view

The most important quantity for making predictions is the predictive distribution $p(\boldsymbol{y}'|\boldsymbol{y})$ or the noiseless predictive distribution $p(\boldsymbol{f}'|\boldsymbol{y})$, where we denote the noiseless predictions by $\boldsymbol{f}'$, i.e. $\boldsymbol{y}' = \boldsymbol{f}' + \boldsymbol{\epsilon}$. To learn that distribution, we have to make modeling assumptions. In the Gaussian process case, the joint model of unseen data $\boldsymbol{f}'$ and seen data $\boldsymbol{y}$ is assumed to be a multivariate Gaussian with mean $\boldsymbol{\mu}$ and a covariance $\boldsymbol{K}$

$$p(\boldsymbol{y}, \boldsymbol{f}') = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K}) = \mathcal{N}\left(\begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{0}\end{pmatrix}, \begin{pmatrix}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}') \\ \boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X}')\end{pmatrix}\right) . \tag{4.2.7}$$

$\boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X})$ is the matrix containing information about the pairwise covariance of the points in $\boldsymbol{X}'$ and $\boldsymbol{X}$, i.e. $\boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X})_{ij}$ is the covariance of $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$.
A huge advantage of a Gaussian distribution is, that the conditional and the marginal distributions of a partitioned Gaussian are analytically tractable and have a very simple closed form (Bishop, 2006). Thus we get the noiseless predictive distribution

$$\boldsymbol{f}'|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{K}') , \tag{4.2.8}$$

where

$$\begin{aligned}\boldsymbol{\mu}' &= \boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X})\left(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{y} \\ \boldsymbol{K}' &= \boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X}') - \boldsymbol{K}(\boldsymbol{X}',\boldsymbol{X})\left(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma^2\boldsymbol{I}\right)^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}') .\end{aligned} \tag{4.2.9}$$

So, the Gaussian process provides a method to predict $\boldsymbol{f}' \in \mathbb{R}^{N'}$ for input locations $\boldsymbol{X}' = (\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{N'})^T$ conditioned on the observed data $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, with inputs $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T$ and outputs $\boldsymbol{y} = (y_1, \ldots, y_N)^T$. Equation 4.2.8 is the closed form joint predictive distribution for $\boldsymbol{X}'$ which is the posterior Gaussian process evaluated at $\boldsymbol{X}'$. The left plot in Figure 4.2.1 shows the mean (in blue) and 2 standard deviations (in light blue in the background) of the prior Gaussian process and some function drawn in different colors. The mean is zero and the kernel function is the RBF $k(x, x') = \exp\left\{-\frac{1}{2}(x - x')^2\right\}$. After observing some data points we get the posterior distribution defined in Equation (4.2.8), which is the prior distribution collapsed at the observed points. Right hand side of Figure 4.2.1 shows the mean and 2 standard deviations of the posterior in light blue and some function drawn in different colors. Note how the uncertainty is reduced in the neighbourhood of the observed data.
Usually the kernel function is dependent on some hyperparamters. The RBF has a variance σ^2 and a lengthscale l parameter that specify the vertical and horizontal scaling of the GP. Hyperparameters are further discussed in Section 4.3. Those parameters are learned from the data by maximizing the data likelihood (also called the marginal likelihood or the evidence) $p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})\mathrm{d}\boldsymbol{f}$. The prior $p(\boldsymbol{f})$ is a GP. Thus, $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$, where $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ and the likelihood is a Gaussian noise model $p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2_{\text{noise}}\boldsymbol{I})$ with mean $\boldsymbol{f}$. In this case, the marginal likelihood has a closed form expression

$$\log p(\boldsymbol{y}) = -\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma^2_{\text{noise}}\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K} + \sigma^2_{\text{noise}}\boldsymbol{I}| - \frac{n}{2}\log 2\pi . \tag{4.2.10}$$

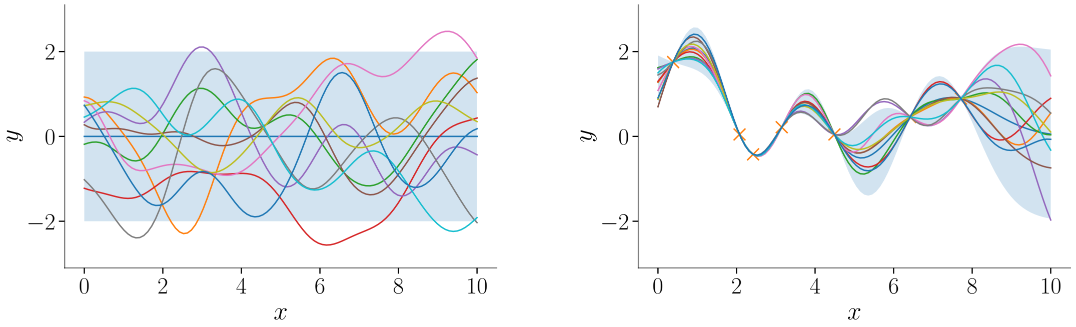


Figure 4.2.1: Visualization of prior and posterior GP for the RBF kernel. Left: Gaussian process prior. Right: Gaussian process posterior after observing 7 data points.

This equation pleasingly separates into an automatically calibrated model fit and complexity term. The data fit term ($\boldsymbol{y}^T(\boldsymbol{K} + \sigma^2_{\text{noise}}\boldsymbol{I})^{-1}\boldsymbol{y}$) penalizes data points lying outside the ellipse given by the covariance matrix. The complexity term ($|\boldsymbol{K} + \sigma^2_{\text{noise}}\boldsymbol{I}|$) is a kind of volume of data sets which are compatible with the model and discourages overcomplex (high volume) models, which are able to explain too many data sets (Rasmussen and Ghahramani, 2001).

The marginal likelihood can be maximized and yields the optimal values for the kernel variance σ^2, kernel length scale l and the noise σ^2_{noise}. A better approach would be to put also a prior on those and infer their posterior. This, however, is not tractable anymore and we have to resort to approximation methods.

So, a Gaussian process is a joint distribution of the whole output space conditioned on the observations $\boldsymbol{y}$ at $\boldsymbol{X}$. It is an infinite dimensional object that can be evaluated at a finite number of points $\boldsymbol{X}' = (\boldsymbol{x}'_1, \ldots, \boldsymbol{x}'_{N'})$ according to Equation (4.2.8). The neglection of all the other points is the same as a marginalization of them. That is due to the marginalization property of a Gaussian

$$\int \mathcal{N}\left(\begin{pmatrix}\boldsymbol{X}_1\\ \boldsymbol{X}_2\end{pmatrix}\middle| \begin{pmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{pmatrix}, \begin{pmatrix}\boldsymbol{K}_{11} & \boldsymbol{K}_{12}\\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22}\end{pmatrix}\right) \mathrm{d}\boldsymbol{X}_2 = \mathcal{N}(\boldsymbol{X}_1|\boldsymbol{\mu}_1, \boldsymbol{K}_{11}) \, . \tag{4.2.11}$$

Each sample from a Gaussian process is a function (an infinite dimensional vector) that we evaluate on N' different points and thus get a N' dimensional vector $\boldsymbol{f}'$. Due to the marginalization property of the Gaussian, it is the same as taking a sample from a multivariate N' dimensional Gaussian distribution.

The function space view is connected to the weight space view. Depending on the kernel we choose, we predefine the set of basis functions that the regression function can use. This connection is given by Mercer's theorem and is introduced in the next section.

4.2.3 Mercer's Theorem

Theorem 4.2.1 *(Mercer) If k is a symmetric and continuous function and the associated*

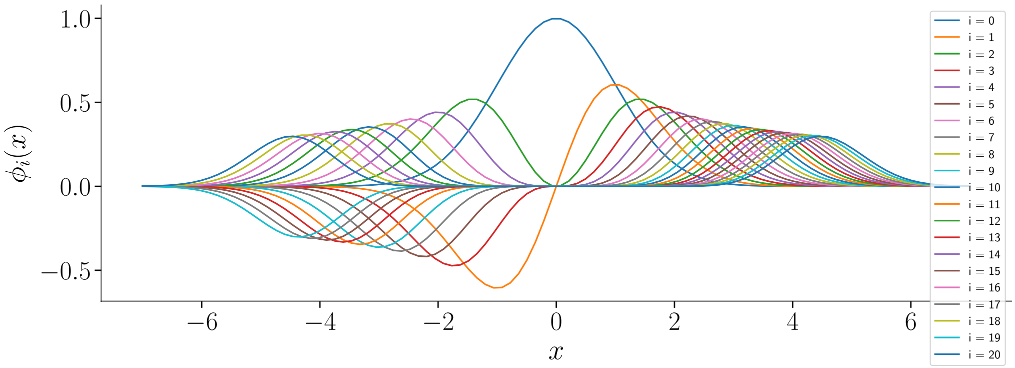


Figure 4.2.2: Basis functions of RBF-kernel (Equation 4.2.13) up to $i = 20$.

matrix $\boldsymbol{K}$ *is positive semi definite, i.e.* $\mathbf{u}^T \boldsymbol{K}\boldsymbol{u} \geq 0 \ \forall \ \boldsymbol{u}$ *then we can write*

$$k(s,t) = \sum_{j=1}^{\infty} \lambda_j \phi_j(s)\phi_j(t) \ , \tag{4.2.12}$$

where $\lambda_j > 0$ *and* ϕ_j *are the eigenvalues and normalized eigenfunctions of* k*. The eigenvalues* λ_i *are absolutely summable, i.e.* $\sum_{i=1}^{\infty} \lambda_i < \infty$ *.*

The theorem states, that there is a kernel function k corresponding to every set of basis functions $\boldsymbol{\phi} = (\phi_1, \phi_2, \dots)$ and also that every valid kernel function k can be decomposed into a set of (possibly infinitely many) basis functions $\boldsymbol{\phi}$. The basis function $\phi(x) = x$ correspond to a linear kernel $k = \alpha x x'$. $\boldsymbol{\phi}(x) = (x, x^2)^T$ corresponds to the quadratic kernel $k(x,x') = xx' + x^2x'^2$. For a given kernel k we can also get back the basis functions. For example: the polynomial kernel $k(x,x') = (xx' + c)^d$ corresponds to all polynomial basis functions up to order d. This concept is really powerful, since we are now also in the position to define a kernel with an infinitely long basis expansion. The RBF kernel $k = \sigma^2 \exp\left\{-\frac{1}{2l^2}(x-x')^2\right\}$, for example, has infinitely many basis functions. It can be decomposed as $k(x,x') = \sum_{i=0}^{\infty} \phi_i(x)\phi_i(x')$ (λ_i is included in the definition of ϕ_i), where

$$\phi_i(x) = \frac{\sigma x^i}{l^i \sqrt{i!}} \exp\left\{-\frac{x^2}{2l^2}\right\} \ . \tag{4.2.13}$$

The basis functions up to $i = 20$ for $\sigma = 1$ and $l = 1$ are shown in Figure 4.2.2.
Gaussian process regression can be viewed as Bayesian linear regression with possibly infinite number of basis functions, where the set of basis functions are the eigenfunctions of the kernel function. Depending on the kernel, the corresponding basis functions and therefore also the regression function will have different properties (i.e.: smoothness, differentiability, periodicity, ...). As we decomposed the RBF kernel into Gaussian basis functions (Equation 4.2.13), we can also decompose it into its Fourier components and get the weights of the frequencies the regression function is composed of. In next section we analyse the Fourier transforms of some of the frequently used kernel functions and show how the kernel function affects the properties of the resulting regression function.

Table 4.1: Popular kernel functions and their Fourier transform. For stationary kernels $r = |x - x'|$.

Name	Functional form	Fourier transform
Linear	$\alpha x x'$	-
RBF	$\sigma^2 \exp\left\{-\frac{r^2}{2l^2}\right\}$	$\sigma^2 l \exp\left\{-\frac{1}{2}l^2\omega^2\right\}$
EXP	$\sigma^2 \exp\left\{-\frac{r}{l}\right\}$	$\sigma^2 \frac{2}{l}/(\frac{1}{l^2} + \omega^2)$
Matern 3/2	$\sigma^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left\{-\frac{\sqrt{3}r}{l}\right\}$	$\sigma^2 \frac{4}{(\sqrt{3}l)^3}/(\frac{1}{3l^2} + \omega^2)^2$
Matern 5/2	$\sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left\{-\frac{5r}{l}\right\}$	$\sigma^2 \frac{16}{3(\sqrt{5}l)^5}/(\frac{1}{5l^2} + \omega^2)^3$
White Noise	$\delta_{x,x'}\sigma^2_{\text{NOISE}}$	1

4.3 Kernel function and its Fourier transformation

In the equation of the mean and the covariance of the posterior GP (Equation 4.2.9) we see that the kernel function is a very important part of the GP regression. The properties of the sampled function f from a GP (i.e. smoothness, periodicity) are determined by the choice of the kernel function. Table 4.1 lists some popular kernel functions. To be a valid kernel function, k has to be a symmetric positive semi-definite function. Only then the resulting matrix will be a valid covariance matrix.

If the kernel only depends on the distance between the two inputs, the kernel is called a stationary kernel. It has a variance and a length scale parameter. The variance parameter is responsible for the variance of the function value $f(x)$ at some point x. It is a vertical rescaling of the kernel as well as of the GP samples. The length scale is also a rescaling parameter but this time for the input scale. Higher length scales will lead to function samples that do not vary much as a function of the input.

Stationary kernels can also be easily proved to be positive semi definite using Bochners theorem.

Theorem 4.3.1 *(Bochner) A continuous stationary function $k(x, x') = \tilde{k}(|x - x'|)$ is positive definite if and only if $\tilde{k}$ is the Fourier transform of a finite positive measure*

$$\tilde{k}(t) = \int \exp\{-i\omega t\}\, d\mu(\omega) \; . \tag{4.3.1}$$

The proof of the theorem is given by Gikhman and Skorokhod (1974). The density $S(\omega)$ corresponding to the measure μ, if exists, is called the spectral density or the power spectrum corresponding to k and the following relationship holds

$$
\begin{aligned}
\tilde{k}(t) &= \frac{1}{2\pi}\int \exp\{i\omega t\}\, S(\omega) \mathrm{d}\omega \\
S(\omega) &= \int \exp\{-i\omega t\}\, \tilde{k}(t) \mathrm{d}t\ .
\end{aligned} \tag{4.3.2}
$$

$S(\omega)$ has a nice interpretation. Fourier transformation is a decomposition of the function f into its Fourier component with different frequencies ω. $S(\omega)$ is the weight of frequency ω in f. Table 4.1 shows the corresponding Fourier transformation of each stationary kernel. The length scale l in the input space $\mathcal{X}$ is like a bandwidth in the Fourier space. For the RBF kernel, for example, the smaller l is, the higher the variation of the samples. This is equivalent to allowing higher frequencies (smaller l increases the width of the RBF Fourier transform).

The Fourier transformation provides a great way to construct valid kernel functions. To construct a valid stationary kernels, we just need to define a density $S(\omega)$ on the Fourier space and the corresponding Fourier transformation will be a kernel function. A very flexible kernel is constructed e.g. by using a mixture of Gaussians in the Fourier space. Wilson and Adams (2013) did exactly that. The resulting kernel is called the spectral mixture kernel.

Figure 4.3.1 shows the form of the kernel function (left) and their corresponding Fourier transformation (right) for kernels listed in Table 4.1 for $\sigma^2 = 1$ and $l = 1$. Fourier space is normalized such that all functions begin at 1. The bottom row shows the same as the top row, the only difference being the log scale at the bottom. Samples from the GP with these kernel functions are shown in Figure 4.3.2. Note the connection of the Fourier transform and the smoothness/differentiability of the sampled functions. The RBF functions are infinity often differentiable and have a double exponentially falling support in the frequency domain (higher frequencies are strongly suppressed). The Ornstein-Uhlenbeck process on the other hand is not differentiable at all. This is due to the high support of higher frequencies, which only fall of with $\frac{1}{\omega^2}$. Plotting the Fourier transform using the log scale (bottom right plot of Figure 4.3.1) makes the mentioned difference in the tails clearly visible. Here we see the strong suppression of high frequencies of the RBF kernel (blue line), whereas the exponential kernel has a very fat tail (red line). The other Matern kernels are somewhere in between those two extremes.

Also the learned function (the GP posterior) very much depend on the form of the kernel function. This is illustrated in Figure 4.3.2, where the left column shows samples from the prior GP and the right columns shows samples from the learned GP posterior after observing some data points.

4.4 Computational complexity and approximations

Gaussian processes are really nice to work with and have the big advantage of being analytically tractable and providing the full uncertainty (i.e. analytically tractable predictive

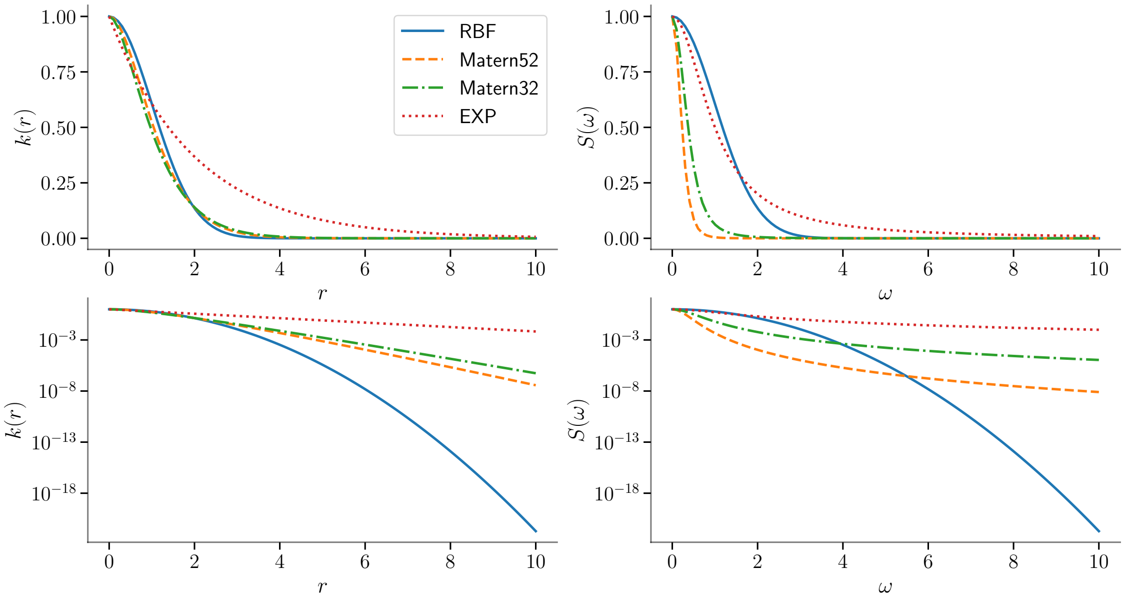


Figure 4.3.1: Popular stationary kernel functions (left) and the corresponding Fourier transformation (right). Bottom row shows the same as the top row. Just the scale of the ordinate is different. Using the log scale makes the differences in the tails more obvious.

distribution). Learning is "just" the application of Equation (4.2.9). However, the evaluation of these equations is computationally quite expensive. The biggest problem is the storage cost and the inversion of the covariance matrix $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$. The storage cost is of order $\mathcal{O}(N^2)$ and the inversion $\mathcal{O}(N^3)$, which does not allow for more than a few thousand data points.

Fortunately, there are many inducing point approximations out there, which reduce the cost of the matrix inversion of $\mathcal{O}(N^3)$ to $\mathcal{O}(M^3)$. The total cost of inference is reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(NM^2)$ and the cost of storage from $\mathcal{O}(N^2)$ to $\mathcal{O}(NM)$. These methods approximate the true Gaussian process by focusing on a smaller number ($M \ll N$) of (pseudo) data points and thus only need to invert a $M \times M$ covariance matrix.

One of the first approximation people come up with was the Nyström approximation to $\boldsymbol{K}$ (Rasmussen and Williams, 2005; Schölkopf and Smola, 2002). Instead of inverting $\boldsymbol{K} \in \mathbb{R}^{N \times N}$, a low rank approximation is made, by using the eigenvalue decomposition of $\boldsymbol{K}$ and only keeping the greatest M instead of all N eigenvalues and eigenvectors.

Snelson and Ghahramani (2006) introduced another method called the fully independent training conditional (FITC), that augments the data distribution $p(\boldsymbol{y})$ with pseudo data (also called inducing points) $\boldsymbol{u}$ and works on the joint distribution $p(\boldsymbol{y}, \boldsymbol{u})$. The joint distribution is assumed to have a sparse structure of the form $p(\boldsymbol{y}, \boldsymbol{u}) = p(\boldsymbol{u}) \prod_i p(y_i|\boldsymbol{u})$, i.e. the data points become independent conditioned on the inducing points. The number of the inducing points is smaller than the number of data points $\boldsymbol{y}$, which reduces the cost of computing the likelihood to only $\mathcal{O}(NM^2)$, where M is the number of the inducing

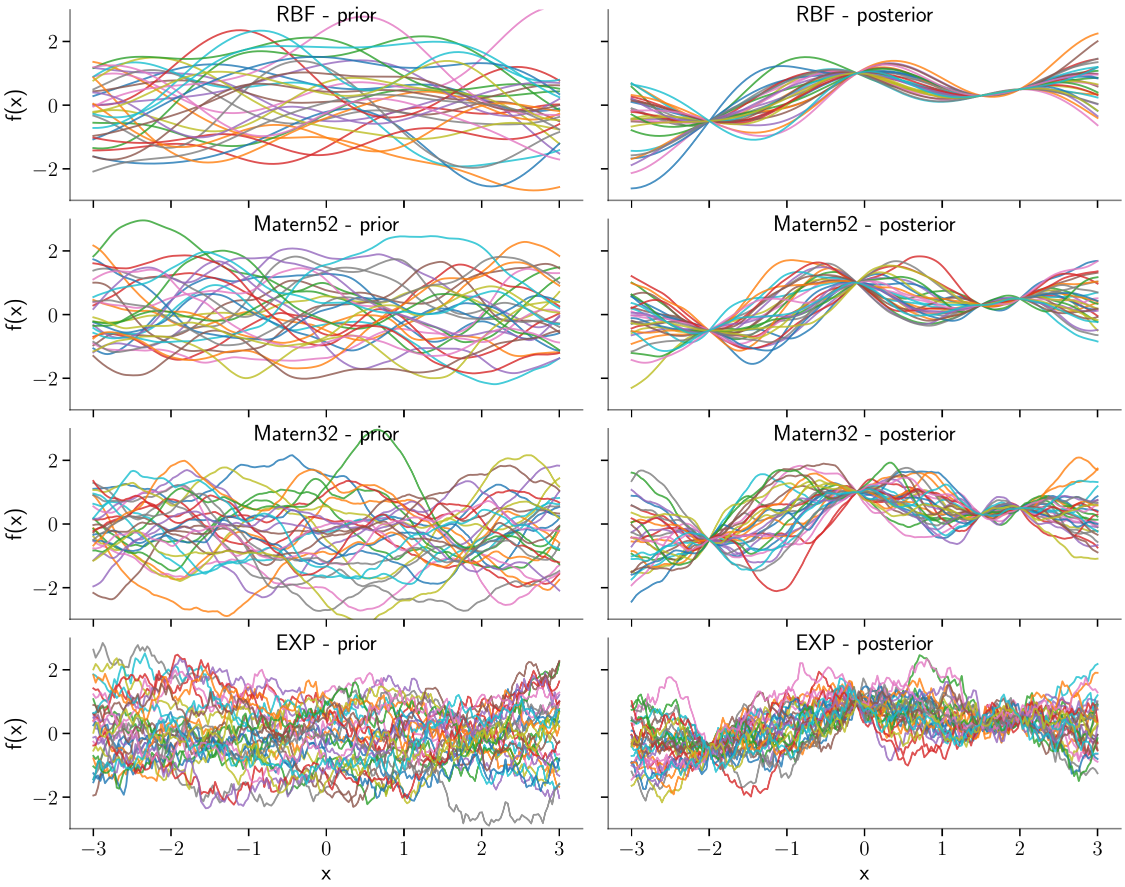


Figure 4.3.2: Samples from GPs. Each row shows samples for different kernel functions. Left: samples from GP prior. Right: samples from GP posterior after observing four point.

points. The introduction of the inducing points makes the model parametric again, where the location of the inducing points become the parameters. They can be jointly learned with the hyperparameters of the model by optimizing the marginal likelihood.

The most successful approximation so far was introduced by Titsias (2009). This approximation belongs to the variational free energy methods. The idea is to approximate the full GP with another variational GP by minimizing their Kullback Leibler divergence. The variational GP tries to approximate the true GP with fewer data points (inducing points). The benefit of this variational ansatz, in contrast to FITC, is that the inducing points are no longer parameters of the model, but variational parameters of the variational approximation. The best fit (in terms of the KL-divergence) is the one where the variational GP resembles the true GP. Therefore, the inducing points in this ansatz are variationally protected against overfitting.

Burt et al. (2019) looked at the computational complexity of variational sparse Gaussian processes introduced by Titsias (2009), to get an arbitrarily good approximation to the exact GP regression. They put an upper bound on the KL divergence from the approximate posterior to the exact posterior and looked at their dependency on the number of inducing points used. Their findings are that the number of inducing points need to be of order $\mathcal{O}((\log N)^D)$ to achieve an error $\mathrm{KL}(q\|p) \leq \epsilon/\delta$ with a probability $1-\delta$ for a fixed ϵ. This is the result for an RBF kernel, if the input to the GP is of bounded support. That means that the number of inducing points increase only as a logarithm of the total number of data points, which is really powerful for very large data sets.

Chapter 5

Bayesian latent variable models

5.1 Introduction

Latent (hidden) variables are variables that are never observed but they, in some sense, generate the observed data. So, if $\boldsymbol{x} \in \mathbb{R}^Q$ are Q latent variables, then the observed data $\boldsymbol{y} \in \mathbb{R}^D$ would be related to the latent variable by some transformation $\boldsymbol{f}$

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon} \,, \tag{5.1.1}$$

where $\boldsymbol{y}$ is the noisy observation with noise $\boldsymbol{\epsilon} \in \mathbb{R}^D$. This modeling choice makes the assumption, that the noiseless data, observed in the D dimensional space are actually living on a Q dimensional manifold. The manifold is described by the function $\boldsymbol{f}$. From that manifold the data is then "lifted up" to D dimensions by the noise. Latent variable models (LVMs) are used to find that Q dimensional linear or nonlinear manifold in the D dimensional space without knowing the actual dimension and the functional representation of it.

Example

An example would be a marionette, that is controlled by a persons hand and thus has very limited degrees of freedom. Let's call the degrees of freedom Q and collect their value in a vector $\boldsymbol{x} \in \mathbb{R}^Q$. We as a observer of a marionette theatre only see the complex movements of the marionette. We can now try to infer the hidden forces that are applied to make the marionette move in a certain way. For that we have to describe the observation somehow. We can choose to measure the position of the head, legs, hands, angle between the lower and upper arm and much more. Each measurement consists of these D different parts which we collect in a vector $\boldsymbol{y} \in \mathbb{R}^D$. We are measuring by some physical device and since they are not perfect, our measurement will be slightly different from the actual state of the marionette, which is a mapping of the hand of the person $\boldsymbol{x}$ to the true position $\boldsymbol{f}(\boldsymbol{x})$ (hand, leg, angle, ...). So at a particular time, the state of the hand $\boldsymbol{x}$ is mapped to the noisy measurement $\boldsymbol{y}$. By observing D time frames, we get N times D

measurements $\boldsymbol{Y} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_N)^T \in \mathbb{R}^{N\times D}$, coming from N different positions of the hand $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)^T \in \mathbb{R}^{N\times Q}$, mapped via $\boldsymbol{f} : \mathbb{R}^Q \to \mathbb{R}^D$ and disturbed by some noise $\boldsymbol{\epsilon}$ [1]. In mathematical terms the generative model is the following: we have a distribution $p(\boldsymbol{x})$ on the latent space $\boldsymbol{x}$. We sample an $\boldsymbol{x}$ and map it to the observed space $\boldsymbol{y}$ via $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon}$. The interesting quantity for a Bayesian is the data distribution $p(\boldsymbol{y})$, i.e. the distribution of all my (seen and unseen) data, the density in the data space. So, given observed data $\boldsymbol{Y} = (\boldsymbol{y}_1, \dots \boldsymbol{y}_N)^T$, we want to come up with a model that maximizes the data distribution $p(\boldsymbol{Y})$ and hopefully is an accurate description also of the data that we have not seen yet. Following the basic laws of probability (the sum and the product rule) we can derive the full generative model. Particularly we are interested in what potentially is out there (what we would observe next) based on the already seen data. The distribution for "what we would observe next" (denoted by $\boldsymbol{y}'$) conditioned on the data we already observed (denoted by $\boldsymbol{Y}$) is called the predictive distribution $p(\boldsymbol{y}'|\boldsymbol{Y})$. To describe this quantity $p(\boldsymbol{y}'|\boldsymbol{Y})$ we introduce auxiliary variables and their causal relationship to the data (the model)[2]

$$p(\boldsymbol{y}'|\boldsymbol{Y}) = \int p(\boldsymbol{y}'|\boldsymbol{f}, \boldsymbol{X}) p(\boldsymbol{f}, \boldsymbol{X}|\boldsymbol{Y}) \mathrm{d}\boldsymbol{f} \mathrm{d}\boldsymbol{X} \ , \tag{5.1.2}$$

where the first term in the integrand is the likelihood of new points $p(\boldsymbol{y}'|\boldsymbol{f}, \boldsymbol{X})$ and the second term $p(\boldsymbol{f}, \boldsymbol{X}|\boldsymbol{Y})$ is the posterior, that puts more weight on the space that is more likely to cause $\boldsymbol{Y}$ and less weight on the space that would have cause very different data than $\boldsymbol{Y}$. The posterior is given by Bayes rule

$$p(\boldsymbol{f}, \boldsymbol{X}|\boldsymbol{Y}) = \frac{p(\boldsymbol{Y}|\boldsymbol{f}, \boldsymbol{X}) p(\boldsymbol{f}, \boldsymbol{X})}{p(\boldsymbol{Y})} \ , \tag{5.1.3}$$

where

$$p(\boldsymbol{Y}) = \int p(\boldsymbol{Y}|\boldsymbol{f}, \boldsymbol{X}) p(\boldsymbol{f}, \boldsymbol{X}) \mathrm{d}\boldsymbol{f} \mathrm{d}\boldsymbol{X} \tag{5.1.4}$$

is the data distribution. Different models make different assumptions for the likelihood $p(\boldsymbol{Y}|\boldsymbol{f}, \boldsymbol{X})$ and the prior $p(\boldsymbol{f}|\boldsymbol{X})$. In this chapter we will look into 2 of these models. The first one is a linear latent variable model, the principle component analysis. The second one is its nonlinear generalization, the Gaussian process latent variable model.

[1] Note that we are trying to infer the latent positions $\boldsymbol{X}$, the function $\boldsymbol{f}$ and the noise $\boldsymbol{\epsilon}$. Note, first of all, the separation of our observations into those three terms and then the assumption of how they interplay to cause $\boldsymbol{Y}$. Each of them will pick up different structure of the observations $\boldsymbol{Y}$. This separation is totally arbitrary and is the choice of the modeler. Each of these separations will pick up some structure of $\boldsymbol{Y}$, based on how we allow them to interplay. We can call the modeling assumption of the noise $\boldsymbol{\epsilon}$ around the function values $\boldsymbol{f}(\boldsymbol{X})$ the likelihood and call the assumption of the function $\boldsymbol{f}$ the prior. Every structure is shared among them and different assumptions for $\boldsymbol{f}$ and $\boldsymbol{\epsilon}$ will lead to different structure sharing. The full specification of all the parts and their interplay is called the model. It is not just the likelihood or just the prior. Both are very important and both are necessary to define. It becomes also obvious here, that what we call "structure" (the thing we want to learn) and the noise (what we want to get rid of) only emerge by defining a model. Without the model, nothing can be learned, because there is no discrimination between structure and noise. Without the model, there is not even a definition of structure.

[2] We assume $p(\boldsymbol{y}'|\boldsymbol{f}, \boldsymbol{X}, \boldsymbol{y}) = p(\boldsymbol{y}'|\boldsymbol{f}, \boldsymbol{X})$, meaning that all information we get from the seen data $\boldsymbol{Y}$ about the unseen data $\boldsymbol{y}'$ is contained in $\boldsymbol{f}$ and $\boldsymbol{X}$.

5.2 Principle Component Analysis

5.2.1 Classical PCA

Principle component analysis (PCA) is a model[3] that assumes the data $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$ lie on a Q dimensional linear subspace of $\mathbb{R}^D$ with a Gaussian noise around the subspace. The frequentists solution for solving the PCA model is the Eigen decomposition, that maps the data $\boldsymbol{Y} = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N)^T \in \mathbb{R}^{N\times D}$ to a lower dimensional matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)^T \in \mathbb{R}^{N\times Q}$ via a linear mapping $\boldsymbol{W} \in \mathbb{R}^{D\times Q}$.

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{\epsilon}\ . \tag{5.2.1}$$

The mapping is learned in a way that minimizes the loss of information. Other interpretations of the classical PCA are that the variance of the projected data in the latent space is maximized (it finds orthogonal directions of maximum variance) or that the mean squared loss on the reconstruction is minimized.

5.2.2 Probabilistic PCA

Probabilistic PCA (PPCA) can be viewed as a generative model, which assumes that the generation of the observed D-dimensional data point $\boldsymbol{y}_n$ has only Q degrees of freedom. So, each of the N data points is generated first by selecting a point from the Q-dimensional latent space and is then mapped to the observed space by a linear mapping

$$\boldsymbol{y}_n = \boldsymbol{W}\boldsymbol{x}_n + \boldsymbol{\epsilon}_n\ , \tag{5.2.2}$$

where the noise $\boldsymbol{\epsilon}_n$ is assumed to be from a zero mean Gaussian with variance σ^2, which leads to the conditional distribution

$$p(\boldsymbol{y}_n|\boldsymbol{W}, \boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{y}_n|\boldsymbol{W}\boldsymbol{x}_n, \sigma^2\boldsymbol{I})\ . \tag{5.2.3}$$

The likelihood of the full data set then becomes

$$p(\boldsymbol{Y}|\boldsymbol{W}, \boldsymbol{X}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{y}_n|\boldsymbol{W}\boldsymbol{x}_n, \sigma^2\boldsymbol{I})\ . \tag{5.2.4}$$

In this equation either $\boldsymbol{X}$ or $\boldsymbol{W}$ can be marginalized out analytically by setting a Gaussian prior on either one of them. In PPCA the prior is set on the latent space $\boldsymbol{X}$, which is assumed to be a standard Gaussian $p(\boldsymbol{X}) = \mathcal{N}(\boldsymbol{X}|\boldsymbol{0}, \boldsymbol{I}\otimes\boldsymbol{I})$. The marginal likelihood is

[3] Distinction of PCA as a model and algorithm: PCA as a model makes the assumption that the Q dimensional manifold in the D dimensional data space is linear and the noise around the manifold is Gaussian. The frequentists algorithm, Eigen decomposition (Golub and van Loan, 2013), to find that subspace is also called PCA. It is an algorithm (step-by-step instruction) that minimizes the noise (i.e. the mean squared error).

analytically tractable and has the form

$$\begin{aligned} p(\boldsymbol{Y}|\boldsymbol{W}) &= \int p(\boldsymbol{Y}|\boldsymbol{W},\boldsymbol{X})p(\boldsymbol{X})\mathrm{d}\boldsymbol{X} \\ &= \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{y}_n|\boldsymbol{0},\boldsymbol{W}\boldsymbol{W}^T+\sigma^2\boldsymbol{I}) \ . \end{aligned} \tag{5.2.5}$$

For a specific data set $\boldsymbol{Y}$, Equation (5.2.5) can either be optimized to get the mapping $\boldsymbol{W}^* = \max_{\boldsymbol{W}} p(\boldsymbol{Y}|\boldsymbol{W})$ or it can be solved within the Bayesian framework, where one puts a prior on $\boldsymbol{W}$ and infers the posterior.

It turns out that the optimization of Equation (5.2.5) in a particular way leads to the solution of the classical PCA, where the Eigen decomposition is used. This is discussed in the next section.

5.2.3 Connection of classical PCA and probabilistic PCA

Tipping and Bishop (1999) show how the PPCA solutions is related to the classical PCA solution. Instead of maximizing Equation (5.2.5) with respect to $\boldsymbol{W}$, we reparameterize the equation by the Singular value decomposition (SVD) of $\boldsymbol{W}$. SVD decomposes a matrix $\boldsymbol{W} \in \mathbb{R}^{D\times Q}$ into $\boldsymbol{U} \in \mathbb{R}^{D\times Q}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{Q\times Q}$ and $\boldsymbol{V} \in \mathbb{R}^{Q\times Q}$, such that $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. $\boldsymbol{U}$ and $\boldsymbol{V}$ are both orthogonal matrices and $\boldsymbol{\Sigma}$ is a diagonal matrix. Equation (5.2.2) can now be written as

$$\boldsymbol{y}_n = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x}_n + \boldsymbol{\epsilon}_n \ , \tag{5.2.6}$$

where the new parameters can be easily interpreted. $\boldsymbol{x}_n$ is first rotated via $\boldsymbol{V}$, then scaled via $\boldsymbol{\Sigma}$ and then rotated "into" the data space via $\boldsymbol{U}$. At the end Gaussian noise is added, which leads to the observation $\boldsymbol{y}_n$.

The fact, that the likelihood $p(\boldsymbol{Y}|\boldsymbol{W})$ only depends on the outer product of $\boldsymbol{W}$, is important to understand the connection between classical PCA and PPCA. The outer product $\boldsymbol{W}\boldsymbol{W}^T$ is a symmetric matrix of rank Q, which has only $DQ - \frac{1}{2}Q(Q-1)$ degrees of freedom[4]. But when we optimize the likelihood with respect to all the elements of $\boldsymbol{W}$, we have DQ parameters. The model is clearly overparameterized. The SVD of $\boldsymbol{W}$ leads to the three matrices $\boldsymbol{U}, \boldsymbol{\Sigma}$ and $\boldsymbol{V}$. The DQ parameters are now split over these matrices and we see the redundancy of the latent space rotation matrix $\boldsymbol{V}$ [5], which has $\frac{1}{2}Q(Q+1)$ free parameters. So, reparameterizing $\boldsymbol{W}$ by its SVD components and only optimizing over $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ will break the rotational symmetry and uniquely identify $\boldsymbol{W}$.

Exactly this is done in the classical PCA case, where the optimization over $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ can be done in closed form and the result is given by the Eigen decomposition of $\boldsymbol{Y}\boldsymbol{Y}^T$ (Tipping and Bishop, 1999).

[4] $\mathrm{Dim}(\boldsymbol{W}\boldsymbol{W}^T) = \mathrm{Dim}(\boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^T) = \mathrm{Dim}(\boldsymbol{U}) + \mathrm{Dim}(\boldsymbol{\Sigma}) = DQ - \frac{1}{2}Q(Q+1) + Q = DQ - \frac{1}{2}Q(Q-1)$

[5] The outer product does not depent on $\boldsymbol{V}$: $\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\boldsymbol{U^T}$

5.2.4 Bayesian PCA

In the Bayesian framework, instead of optimizing the parameters, we infer the posterior over those by using Bayes rule

$$p(\boldsymbol{W}|\boldsymbol{Y}) = \frac{p(\boldsymbol{Y}|\boldsymbol{W})\ p(\boldsymbol{W})}{p(\boldsymbol{Y})}\ . \tag{5.2.7}$$

In this case, however, the marginal likelihood $p(\boldsymbol{Y}) = \int p(\boldsymbol{Y}|\boldsymbol{W})p(\boldsymbol{W})\mathrm{d}\boldsymbol{W}$ is not tractable anymore. We have to resort to approximation methods. As we will see in Chapter 8 the rotational symmetry of the posterior, that arises due to the rotational symmetry of the likelihood and the prior[6] will be a problem. The solution to that problem is to reparameterize $\boldsymbol{W}$ again in its SVD components and approximate the posterior of the components instead of $\boldsymbol{W}$ itself. We elaborate more on that in Chapter 8.

5.3 Gaussian process Latent Variable Models

Lawrence (2005) introduced the Gaussian process latent variable model (GPLVM) for nonlinear dimensionality reduction. In his approach, instead of marginalising the latent positions $\boldsymbol{X}$, he marginalized the parameters $\boldsymbol{W}$ [7]. The equation

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}^T + \boldsymbol{\epsilon}\ , \tag{5.3.1}$$

where $\boldsymbol{\epsilon} \in \mathbb{R}^{N \times D}$ and $\boldsymbol{\epsilon}_{i,j} \sim \mathcal{N}(0, \sigma)$ is an independent noise, can be written in 2 ways

$$\boldsymbol{Y}_{n,:} = \boldsymbol{W}\boldsymbol{X}_{n,:} + \boldsymbol{\epsilon}_{n,:} \tag{5.3.2}$$

$$\boldsymbol{Y}_{:,d} = \boldsymbol{X}\boldsymbol{W}_{d,:} + \boldsymbol{\epsilon}_{:,d}\ . \tag{5.3.3}$$

We saw that the marginalization of Equation (5.3.2) leads to Equation (5.2.5), where the rows of $\boldsymbol{Y}$ are modeled as being conditionally independent. In the second case (Equation 5.3.3), however, the columns of $\boldsymbol{Y}$ are modeled as being conditionally independent[8]. Equation (5.3.3) is the dual space analogon to Equation (5.3.2) and the likelihood of that equation becomes

$$p(\boldsymbol{Y}_{:,d}|\boldsymbol{X}, \boldsymbol{W}_{d,:}) = \mathcal{N}(\boldsymbol{Y}_{:,d}|\boldsymbol{X}\boldsymbol{W}_{d,:}, \sigma^2\boldsymbol{I})\ . \tag{5.3.4}$$

The marginalization of the parameters $\boldsymbol{W}$ by assuming a standard Gaussian prior $p(\boldsymbol{W}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ leads to

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{Y}_{:,d}|\boldsymbol{0}, \boldsymbol{K} + \sigma^2\boldsymbol{I})\ , \tag{5.3.5}$$

[6] Note that both, the likelihood and the prior, are not directly a function of $\boldsymbol{W}$ but its outter product $\boldsymbol{W}\boldsymbol{W}^T$. The outter product, however, does not change by a rotation of $\boldsymbol{W}$, i.e. for a rotated $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{R}$ we get $\tilde{\boldsymbol{W}}\tilde{\boldsymbol{W}}^T = \boldsymbol{W}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^T = \boldsymbol{W}\boldsymbol{W}^T$.

[7] Ideally, one would marginalize both, the parameters $\boldsymbol{W}$ and the latent positions. However, this is not tractable anymore.

[8] In the linear case both choices give the same maximum likelihood result (Lawrence, 2005). This is the case, because $\boldsymbol{X}$ and $\boldsymbol{W}$ always appear in a product. Since in both cases we assume a Gaussian prior over the variable we marginalize, it does not matter, which one we marginalize and which one we optimize.

where $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{K}_{ij} = \boldsymbol{X}_{i,:}^T \boldsymbol{X}_{j,:}$. Compared to Equation (4.1.3), this equation can be seen as a Gaussian process with zero mean and a linear kernel function $k(\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:}) = \boldsymbol{X}_{i,:}^T \boldsymbol{X}_{j,:}$. It is a product of linear regressions across the outputs $\boldsymbol{Y}_{:,d}$. The model is generalized by replacing the linear kernel $\boldsymbol{K}_{ij}$ with a nonlinear one. The generalization to the nonlinear model is called the GPLVM. Thus, by choosing a nonlinear kernel for the mapping from the latent to the observed space, Equation (5.3.2), for each of the N data points become

$$\boldsymbol{Y}_{n,:} = \boldsymbol{f}(\boldsymbol{X}_{n,:}) + \boldsymbol{\epsilon}_{n,:} \tag{5.3.6}$$

and for each dimension d

$$\boldsymbol{Y}_{:,d} = f_d(\boldsymbol{X}_{:,d}) + \boldsymbol{\epsilon}_{:,d} \ , \tag{5.3.7}$$

where $\boldsymbol{f} = (f_1, \ldots, f_D)$ is a group of D samples from a GP with kernel function k. By doing this we assume the rows of $\boldsymbol{Y}$ to be jointly Gaussian distributed with covariance given by k and the columns of $\boldsymbol{Y}$ to be independent. For a zero mean Gaussian random noise $\boldsymbol{\epsilon}$ with variance σ^2 and with a GP prior on $\boldsymbol{f} \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{K} \otimes \boldsymbol{I})$, the marginal likelihood of $\boldsymbol{Y}$ becomes

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{Y}_{:,d}|\boldsymbol{0}, \boldsymbol{K}) = \frac{1}{(2\pi)^{ND/2}|\boldsymbol{K}|^{D/2}} \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{K}^{-1}\boldsymbol{Y}\boldsymbol{Y}^T)\right) , \tag{5.3.8}$$

where the elements of the covariance matrix are given by $\boldsymbol{K}_{ij} = k(\boldsymbol{X}_{i,:}, \boldsymbol{X}_{j,:}) + \sigma^2\delta_{ij}$ and δ_{ij} is the delta function. As suggested in Lawrence (2005), we can optimize the marginal likelihood with respect to the latent positions $\boldsymbol{X}$ and the hyperparameters. Ideally, one would also like to marginalize over $\boldsymbol{X}$ by placing a prior over those, instead of optimizing them. This integral, however, is not tractable anymore and one resorts to approximation methods, which are described in Chapter 3. A similar classical method that also nonlinearizes PCA is the kernel PCA (Schölkopf et al., 1998), which kernelizes the data space instead of the latent space as in the case of GPLVM. It then performs PCA in the uplifted data space by the corresponding kernel.

In Chapter 7 we use the GPLVM to estimate the covariance between different data points. The data points will be different financial assets. After inferring the covariance structure between the assets we can build more sophisticated financial models.

Chapter 6

Finance

In this chapter we look into some financial models that potentially can be generalized using Gaussian processes and Gaussian process latent variable models (GPLVMs). We start by introducing linear latent space models that are used for financial modeling and extend those to nonlinear models using the GPLVM in the next section. In this chapter we also introduce the modern portfolio theory (MPT) (Markowitz, 1952) that can be used to build portfolios that maximize returns and minimize risk based on estimates of mean returns of the assets in the portfolio and their correlation. In the experiments section of Chapter 7 we use MPT and GPLVMs to construct low volatility portfolios. The estimates of the covariance are provided by the GPLVMs. There are also other methods to estimate the covariances. Those are also briefly discussed at the end of this chapter.

6.1 Latent space models in finance

In this section we are going to discuss the two main latent space models for modeling the return of financial assets. The first one is the capital asset pricing model (CAPM) and the second model is the extension of the CAPM, called the arbitrage pricing theory (APT). At the end of this section we also introduce Fama-French three-factor model and Fama-French five-factor model.

6.1.1 Capital Asset Pricing Model (CAPM)

How risks and returns of an investment are related has long been a topic for business and research. The CAPM, also called the single-index or the single-factor model (Sharpe, 1964) assumes that the return $\tilde{r}_n$ of a stock n has a linear dependency on its risk β_n and is a noisy observation of the linear function

$$\tilde{r}_n = \alpha_n + r_f + \beta_n(\tilde{r}_m - r_f) + \epsilon_n \; , \tag{6.1.1}$$

where $\tilde{r}_m$ is the return of the market, α is the intercept, r_f is the risk free rate of return, β_n is the volatility of the stock n relative to the market and ϵ_n is some random noise. The single factor β captures the information about the movement of the stock compared to

the market. Stocks that have a higher volatility than the market have a higher β. They have higher risk and therefore also higher return.
In efficient markets the expected value of α is zero. Therefore, we set α to zero and express the return of the stock $\tilde{r}_n$ and the return of the market $\tilde{r}_m$ in terms of the excess-returns $r_n = \tilde{r}_n - r_f$ and $r_m = \tilde{r}_m - r_f$. Thus Equation (6.1.1) becomes

$$r_n = r_m \beta_n + \epsilon_n \ . \tag{6.1.2}$$

This single-index (factor) model couples the returns r_n of a stock n to the market return r_m. The variability in the return r_n not explained by the market r_m is assumed to be noise ϵ_n. There are many improvements to that model which capture more variability. One of them is the multi-index model, which assumes many more factors, that are unobserved (a linear latent space model). Another one, which has observed factors, is for example the Fama-French three-factor model[1]. We look closer into these models in the next sections.

6.1.2 Arbitrage theory of capital asset pricing

Whereas in CAPM a single unobserved factor is interpreted as the market return r_m, Arbitrage theory of capital asset pricing (APT) assumes that there are Q latent factors $\boldsymbol{F} \in \mathbb{R}^{D\times Q}$ on D days (Ross, 1976)[2]. If we denote $\boldsymbol{r} \in \mathbb{R}^{N\times D}$ as the matrix of returns of N stocks on D days and $\boldsymbol{B} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N)^T \in \mathbb{R}^{N\times Q}$ as the couplings of the N stocks to the Q factors, we get

$$\begin{aligned} \boldsymbol{r}_{n,:} &= \boldsymbol{F}\boldsymbol{\beta}_n + \boldsymbol{\epsilon}_n \\ \boldsymbol{r} &= \boldsymbol{B}\boldsymbol{F}^T + \boldsymbol{\epsilon} \ . \end{aligned} \tag{6.1.3}$$

$\boldsymbol{\beta}_n$ are the sensitivities. They specify to which extent the return of stock n is influenced by a particular factor. In the CAPM the one factor $\boldsymbol{r}_m$ is the return of the market. But in the APT, the factors are some latent/unobserved quantities. Note, that Equation (6.1.3) is a linear latent space model and the factors $\boldsymbol{F}$ can be inferred using e.g. PCA. The drawback of PCA is, that it assumes an error with fixed variance for each element of the data/return matrix $\boldsymbol{r}$ (see Equation (5.2.4)). A better approach to find the latent factors is the factor analysis (Everett, 1984), that allows different noise variances for each row (asset) of $\boldsymbol{r}$. Compared to Equation (5.2.4), the likelihood now becomes

$$p(\boldsymbol{r}|\boldsymbol{B}, \boldsymbol{F}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{r}_{:,d}|\boldsymbol{B}\boldsymbol{F}_{d,:}, \boldsymbol{\Psi}) \ , \tag{6.1.4}$$

[1] Fama and French (1993) introduced a three-factor model called the Fama-French model. It improved the CAPM by including two additional factors to the market factor. These two factors are related to the firms size and value. There is even a Fama-French five factor model (Fama and French, 2015) that, in addition to the previous three factors, also includes profitability and investment of a firm.

[2] The factors are sometimes also called indexes. Thus, APT is also refereed to as multi-index model. CAPM on the other hand is called single-index model.

where $\boldsymbol{\Psi} = \mathrm{diag}(\sigma^2_{\mathrm{noise},1}, \ldots, \sigma^2_{\mathrm{noise},N})$ is the diagonal matrix with different noise variances of each asset. Thus, the marginal distribution, where the couplings are marginalized out, becomes

$$p(\boldsymbol{r}|\boldsymbol{B}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{r}_{:,d}|\boldsymbol{0}, \boldsymbol{B}\boldsymbol{B}^T + \boldsymbol{\Psi}) \; . \tag{6.1.5}$$

The optimization of this equation provides the couplings $\boldsymbol{B}$ and the variances of each asset $\boldsymbol{\Psi}$. Unlike in the PPCA, there is no analytic result to the optimization of Equation (6.1.5). Thus it has to be solved iteratively. Bishop (2006) provides the steps for the optimization using the EM (expectation maximization) algorithm. The factors $\boldsymbol{F}$ can be calculated using

$$\mathbb{E}[\boldsymbol{F}_{d,:}] = \boldsymbol{G}\boldsymbol{B}^T\boldsymbol{\Psi}^{-1}\boldsymbol{r}_{:,d} \tag{6.1.6}$$

$$\boldsymbol{G} = (\boldsymbol{I} + \boldsymbol{B}^T\boldsymbol{\Psi}^{-1}\boldsymbol{B})^{-1} \; . \tag{6.1.7}$$

The solution to Equation (6.1.5) provides the covariance matrix $\boldsymbol{K} = \boldsymbol{B}\boldsymbol{B}^T + \boldsymbol{\Psi} \in \mathbb{R}^{N \times N}$ between the N different assets. This matrix can now be used to calculate optimal portfolios using modern portfolio theory (Section 6.2).

6.1.3 Fama-French three-factor model

Yet again another extension of the CAPM is the Fama-French three-factor model Fama and French (1993). In addition to the market risk, the Fama-French three-factor model also includes two other factors. The first one is the SMB (small minus big) which measures the historic excess of small-cap companies over big-cap companies. The reasoning behind this factor is that in the long-term, small companies will have higher returns than larger companies. The other factor is the HML (high minus low) and represents the spread in returns between companies with a high book-to-market value ratio and companies with a low book-to-market value ratio. The reasoning behind this factor is that in the long term companies with high book-to-market value will outperform companies with low book-to-market value. With these factors the CAPM can be rewritten as

$$\tilde{r}_n = \alpha_n + r_f + \beta_n(\tilde{r}_m - r_f) + s_n\mathrm{SMB} + h_n\mathrm{HML} + \epsilon_n \; . \tag{6.1.8}$$

The three-factor model is a significant improvement over the CAPM and adjusts for outperformance tendencies. But also this model has its drawbacks and overlooks a lot of the variability in average returns which are related to profitability and investment. Therefore, the three-factor model was extended to the five-factor model by Fama and French (2015). The first addition is the RMW (robust minus weak) that tries to distinguish between the most profitable and the least profitable companies and the second addition is the CMA (conservative minus aggressive) that tries to distinguish between companies that invest conservatively and companies that invest more aggressively. Equation (6.1.8) is then extended to

$$\tilde{r}_n = \alpha_n + r_f + \beta_n(\tilde{r}_m - r_f) + s_n\mathrm{SMB} + h_n\mathrm{HML} + m_n\mathrm{RMW} + c_n\mathrm{CMA} + \epsilon_n \; . \tag{6.1.9}$$

In the Fama-French model the factors are observed and it is strictly speaking not a latent variable model.

6.2 Modern Portfolio Theory

Markowitz (1952) provided the foundation for the modern portfolio theory (MPT), for which he received a Nobel Prize in economics. MPT provides a way to manage a portfolio and is based on the so called risk-return relationship. In the CAPM, for example, we saw that higher risks β_n lead to higher returns (Equation 6.1.2). It turns out that we can reduce the risk of our investments without reducing the returns. To understand that, let's assume we have two assets with the same risk profile β and, according to the CAPM, also the same return. By holding only one of them, our risk is β. But if the stocks have a correlation $\sigma < 1$, by combining them, we can reduce our risk, without losing returns. The holding of different stocks, which do not perfectly correlate, is called diversification.

MPT provides a basic and simple way for diversification based on the level of risk an investor seeks. The aversion or the desire for risk of an investor is encoded in the risk tolerance q. Given N different assets and the risk tolerance q of the investor, MPT provides a formula to calculate the optimal weights

$$\boldsymbol{w}_{\text{opt}} = \min_{\boldsymbol{w}} \left(\boldsymbol{w}^T \boldsymbol{K} \boldsymbol{w} - q \hat{\boldsymbol{r}}^T \boldsymbol{w} \right) \ , \tag{6.2.1}$$

where $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is the covariance and $\hat{\boldsymbol{r}} \in \mathbb{R}^N$ is the mean return of the N different assets. Since $\hat{\boldsymbol{r}}$ is very hard to estimate in general and we are primarily interested in the estimation of the covariance matrix $\boldsymbol{K}$, we set q to zero and get the minimal risk portfolio (for an investor with very high aversion for risk $q = 0$)

$$\boldsymbol{w}_{\text{opt}} = \min_{\boldsymbol{w}} \left(\boldsymbol{w}^T \boldsymbol{K} \boldsymbol{w} \right) \ . \tag{6.2.2}$$

The portfolio for assets with the optimal weights $\boldsymbol{w}_{\text{opt}}$ according to Equation (6.2.2) is called the minimal risk portfolio. It minimizes the overall risk assuming the estimated $\boldsymbol{K}$ is the true covariance.

In the section on portfolio allocation 7.3.1, we will use the GPLMV, as described in Section 7.1, to estimate the covariance matrix $\boldsymbol{K}$. $\boldsymbol{K}$ can be used in Equation (6.2.2) to get the optimal weights $\boldsymbol{w}_{\text{opt}}$ for the assets in the portfolio.

6.3 Estimation of covariance matrices

In this section we want to answer the question of "how to estimate the covariance matrix of N assets if their returns are given on D days", i.e. how do we get $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ given the return matrix $\boldsymbol{r} \in \mathbb{R}^{N \times D}$. Based on that estimation we can build a minimal risk portfolio according to MPT (Equation 6.2.2).

Sample covariance matrix

A simple and easy way to calculate an estimation of the true covariance matrix is the sample covariance matrix. The sample covariance matrix (empirical covariance) of the N stocks in $\boldsymbol{r}$ is given by

$$\boldsymbol{K} = \frac{1}{D}(\boldsymbol{r} - \hat{\boldsymbol{\mu}})(\boldsymbol{r} - \hat{\boldsymbol{\mu}})^T \ , \tag{6.3.1}$$

where $\hat{\boldsymbol{\mu}}_n = \frac{1}{D}\sum_{d=1}^{D} \boldsymbol{r}_{nd}$ is the mean of the samples. This estimation is also called the classical maximum likelihood estimator provided the number of observations D is large enough compared to the number of assets N. The maximum likelihood estimator provides an unbiased estimation (i.e. the expected value is equal to the true covariance matrix). However, if the number of observations D is not large enough compared to the number of assets N, the sample covariance is known to be very unstable and can even become singular. To cope with this problem, a wide range of estimators have been developed and employed in portfolio optimization.

Ledoit-Wolf estimation

One of the methods to counteract the drawbacks of the sample covariance matrix is the class of shrinkage estimators. The Ledoit-Wolf estimation is a shrinkage estimation of the sample covariance matrix $\boldsymbol{S}$ to a more structured matrix $\boldsymbol{F}$ (called the shrinkage target)

$$\boldsymbol{K} = \delta \boldsymbol{S} + (1 - \delta)\boldsymbol{F} \ , \tag{6.3.2}$$

where δ is called the shrinkage coefficient. Ledoit and Wolf (2004) propose a formula to compute the optimal shrinkage coefficient that minimizes the mean squared error between the shrinkage estimator and the true covariance matrix. In our experiments we use the implementation in the Python toolbox scikit-learn (Pedregosa et al., 2011), where the sample covariance is shrunk towards the identity matrix

$$\boldsymbol{K} = \delta \frac{\mathrm{Tr}\boldsymbol{S}}{D} \boldsymbol{S} + (1 - \delta)\boldsymbol{I} \ , \tag{6.3.3}$$

where $\mathrm{Tr}\boldsymbol{S}$ is the trace of $\boldsymbol{S}$ and D is the number of observations.

In the next chapter we are using GPLVMs to estimate the covariance matrix of different assets. We then compare the results we get using the GPLVMs estimates to the results we get when we use the sample covariance matrix and the Ledoit-Wolf estimation.

Chapter 7

Gaussian process latent variable models in finance

In this chapter we build upon the financial models introduced in Chapter 6. We start with a nonlinear extension of the arbitrage theory of capital asset pricing (APT) introduced in Section 6.1.2. Here, we show that the Gaussian process latent variable model (GPLVM) is reduced to APT when using a linear kernel. After that we conduct experiments and report results on how to use GPLVMs when modeling financial data.
This chapter is an extension of the published work in the Advances in Intelligent Systems and Computing (Nirwan and Bertschinger, 2019b).

7.1 Nonlinear extension of APT using GPLVMs

In Section 5.3 we introduced the GPLVM as a nonlinear extension of the dual PCA and in Section 6.1.2 we introduced the APT. Equation (6.1.3) shows the general form of the APT which is quite similar to what is done in PCA. Given the observed data matrix $\boldsymbol{r} \in \mathbb{R}^{N\times D}$, we try to infer the latent positions $\boldsymbol{B} \in \mathbb{R}^{N\times Q}$ (also called the couplings) and the mapping $\boldsymbol{F} \in \mathbb{R}^{D\times Q}$ (also called latent factors)[1]. The assumption is that $\boldsymbol{F}$ maps the latent positions $\boldsymbol{B}$ to the data space linearly. We generalize the model by allowing for nonlinear mapping

$$\boldsymbol{r}_{:,d} = f_d(\boldsymbol{B}_{:,d}) + \boldsymbol{\epsilon}_{:,d} \ , \tag{7.1.1}$$

where $f_d \sim \mathcal{GP}(0,k)$, with a nonlinear kernel function k and $\boldsymbol{\epsilon}_{:,d} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$, with $\boldsymbol{\Psi} = \operatorname{diag}(\sigma^2_{\text{noise},1}, \ldots, \sigma^2_{\text{noise},N})$. Note that the inclusion of a nonisotropic noise variance $\boldsymbol{\Psi}$, allows each asset to have its own independent noise variance σ_n^2. Note also, that by choosing one of the stationary kernel functions from Table 4.1, we do not allow different assets to have their own variance, which is not a good idea for financial assets, since they have different volatilities. Therefore, in the case of a stationary kernel we decompose our covariance matrix $\boldsymbol{K}_{\text{cov}}$ into a vector of coefficient scales $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)^T$ and a

[1] The namings (coupling or latent positions for $\boldsymbol{B}$ and mapping or latent factors for $\boldsymbol{F}$) are totally arbitrary. In the first place, they are just a decomposition of the data matrix $\boldsymbol{Y}$.

Table 7.1: Return of four different stocks on four different days.

	24.10.19	25.10.19	28.10.19	29.10.19
AAPL	0.16	1.23	1.00	-2.31
GOOGL	0.11	0.41	1.95	-2.19
AMZN	1.05	-1.09	0.89	-0.80
MSFT	1.96	0.56	2.45	-0.94

correlation matrix $\boldsymbol{K}_{\text{corr}}$, such that $\boldsymbol{K}_{\text{cov}} = \boldsymbol{\Sigma}\boldsymbol{K}_{\text{corr}}\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is a diagonal matrix with $\boldsymbol{\sigma}$ on the diagonal. So, for any stationary kernel function k_{st}, the kernel matrix $\boldsymbol{K}_{\text{st}}$ becomes

$$\boldsymbol{K}_{\text{st}} = \boldsymbol{\Sigma}\ k_{\text{st}}(\boldsymbol{B}, \boldsymbol{B})\ \boldsymbol{\Sigma} + k_{\text{noise}}(\boldsymbol{B}, \boldsymbol{B})\ , \tag{7.1.2}$$

and each element of $\boldsymbol{K}_{\text{st}}$ has the form

$$(\boldsymbol{K}_{\text{st}})_{ij} = \sigma_i \sigma_j k_{\text{st}}(\boldsymbol{B}_{i,:}, \boldsymbol{B}_{j,:}) + \delta_{ij}\sigma^2_{\text{noise},i}\ . \tag{7.1.3}$$

Since the variance σ^2, which was in the definition of k_{st}, is now outside the function, we can set $\sigma^2 = 1$ in all stationary kernels in Table 4.1. In the case of the linear kernel we do not have to model the variance separately and still have

$$\boldsymbol{K}_{\text{linear}} = k_{\text{linear}}(\boldsymbol{B}, \boldsymbol{B}) + k_{\text{noise}}(\boldsymbol{B}, \boldsymbol{B})\ , \tag{7.1.4}$$

because the scale of $\boldsymbol{B}_{n,:}$ can pick up the variance of the n-th asset.
As described in Section 5.3, the likelihood of the full model is given by

$$p(\boldsymbol{r}|\boldsymbol{B}, \boldsymbol{\theta}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{r}_{:,d}|\boldsymbol{0}, \boldsymbol{K}) = \frac{1}{(2\pi)^{ND/2}|\boldsymbol{K}|^{D/2}} \exp\left(-\frac{1}{2}\text{tr}(\boldsymbol{K}^{-1}\boldsymbol{r}\boldsymbol{r}^T)\right)\ , \tag{7.1.5}$$

where the couplings $\boldsymbol{B}$ and the kernel hyperparamters $\boldsymbol{\theta}$ are included in $\boldsymbol{K}$ which for stationary kernels is given by Equation (7.1.2) and for the linear kernel by Equation (7.1.4). The inference of the posterior of the couplings $\boldsymbol{B}$ and the kernel hyperparameters $\boldsymbol{\theta}$ is analytically not tractable anymore. In the experiments in Section 10.4 we use variational inference and approximate the posterior $p(\boldsymbol{B}, \boldsymbol{\theta}|\boldsymbol{r})$ by independent Gaussians by maximizing the ELBO.

7.2 Modeling and data collection

We collect the stock prices of the stocks from the S&P500, whose daily close prices were available for the whole training period. The data can be downloaded from Yahoo Finance. The data of N stock on $D+1$ days are collected in a matrix $\boldsymbol{p} \in \mathbb{R}^{N\times(D+1)}$. Left hand side of Figure 7.2.1 shows the normalized price series (price divided by the starting price,

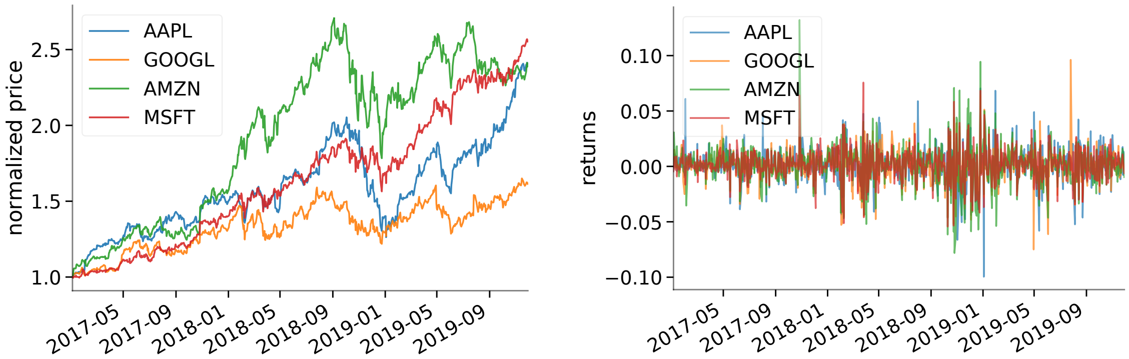


Figure 7.2.1: Price series (left) and return series (right) of four stocks over two years.

such that every series starts a 1.0) for four different stock over two years. The returns are the change of the price from one day to the next and are collected in the return matrix $\boldsymbol{r} \in \mathbb{R}^{N \times D}$, where each element of the matrix is given by

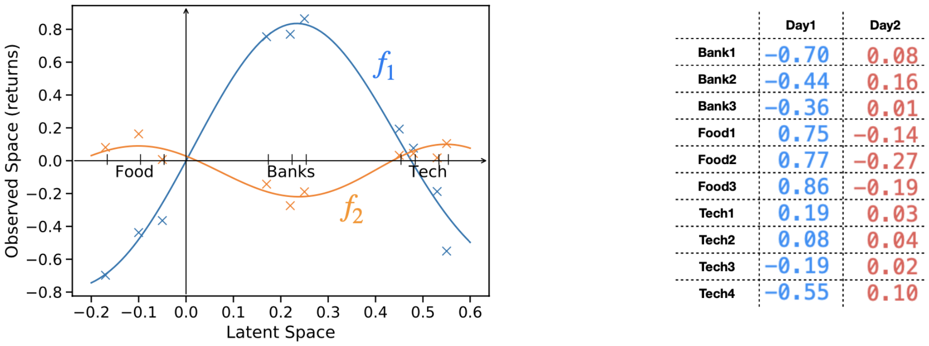
$$\boldsymbol{r}_{nd} = \frac{\boldsymbol{p}_{n,d} - \boldsymbol{p}_{n,d-1}}{\boldsymbol{p}_{n,d-1}} \, . \tag{7.2.1}$$

The returns are also shown on the right hand side of Figure 7.2.1. The returns look a lot like noise. This is indeed the case. It is not purely noise but the correlation between different columns is almost zero[2]. This observation is included in our model by assuming the likelihood of the GPLVM being a product of likelihoods for the returns for different days d. Even though there is no structure in the column, there is a lot of structure in the rows. Table 7.1 shows the return matrix and here we see that different rows are positively correlated. If the return of one stock is positive at a particular day, most likely the returns of other similar stocks will be positive too. This is the structure that we want to pick up by the GPLVM, which assumes a multivariate Gaussian relationship within the rows (stocks).

The question is: Can we come up with a generative model, that generates data with that structure, where the rows are correlated but the columns are independent?[3] Figure 7.2.2 (left hand side) shows the generative process. We assume, that the stocks are embedded in a latent space, which is the x-axis. Similar stocks are closer to each other (this corresponds to higher correlation for stationary kernel functions as we saw in Chapter 4). Y-axes corresponds to the observed space, which are the returns in this case. Each day a random function is drawn from an unknown distribution, that maps the latent space to the observed space. On day one, for example, the blue function is drawn and on another day the red

[2] Higher correlation would imply, that there is some structure in the change of the price from today to tomorrow. This information would mean, that given the price/return of today, we have some information for the price for tomorrow (we can forecast the price for tomorrow). We could trade on this information and as soon as we start trading the pattern will disappear. See (Fama, 1970) on efficient markets.

[3] The following explanation can be seen as an alternative point of view that does not explain the use of GPLVM by the generalization of APT (as in Section 7.1), but results into the same model.



	Day1	Day2
Bank1	−0.70	0.08
Bank2	−0.44	0.16
Bank3	−0.36	0.01
Food1	0.75	−0.14
Food2	0.77	−0.27
Food3	0.86	−0.19
Tech1	0.19	0.03
Tech2	0.08	0.04
Tech3	−0.19	0.02
Tech4	−0.55	0.10

Figure 7.2.2: Generative process of generating a matrix with independent columns but certain structure within each column (mimicking the return matrix $\boldsymbol{r}$).

one. The observed noisy values are the returns on those days. The table on the right hand side of Figure 7.2.2 shows the values corresponding to the blue and the red functions. This generative process outputs the structure that we want to capture in our financial data.
We want to revert this process. We observe only the return matrix $\boldsymbol{r}$. Our task now is to infer the unknown latent space and the function distribution that resulted into that particular observation $\boldsymbol{r}$. In our case we use Gaussian processes to model the unknown function distribution and combine that with latent variable models, which results in the GPLVM. The likelihood for the observation $\boldsymbol{r}$ is given by Equation (7.1.5) and we assume the following priors for the unknowns

$$\begin{aligned} &\boldsymbol{B} \sim \mathcal{N}(0,1) \\ &l, \sigma_{\text{lin}} \sim \text{InvGamma}(3,1) \\ &\boldsymbol{\sigma}, \sigma_{\text{noise}} \sim \mathcal{N}(0,0.5)\ . \end{aligned} \tag{7.2.2}$$

The latent space $\boldsymbol{B}$ is assumed to be a standard normal. If we increase the latent space dimensions, the standard normal will allocate more and more volume for the latent positions, which will lead to overfitting. To counter that, we set an inverse Gamma prior on the length scale l and σ (σ^2 is the variance of the linear kernel, which has a similar functionality as l in the stationary kernels). It allows the length scale to take on high values easily and therefore shrinks the effective latent space volume. The inverse Gamma prior also suppresses very small values for the length scale and by this also counteracts overfitting. The kernel standard deviation σ_{noise} and $\boldsymbol{\sigma} = \text{diag}(\boldsymbol{\Sigma})$ are assigned a half Gaussian prior with variance 0.5, which is essentially a flat prior, since the returns are rarely above 0.1 for a day.

7.2.1 Model comparison

There are a lot of ways to evaluate the fit of the data $\boldsymbol{r}$ with the GPLVM. Since the GPLVM projects the data from a D-dimensional data space to the Q-dimensional latent space, we can look at the reconstruction error. One measure for the reconstruction error is

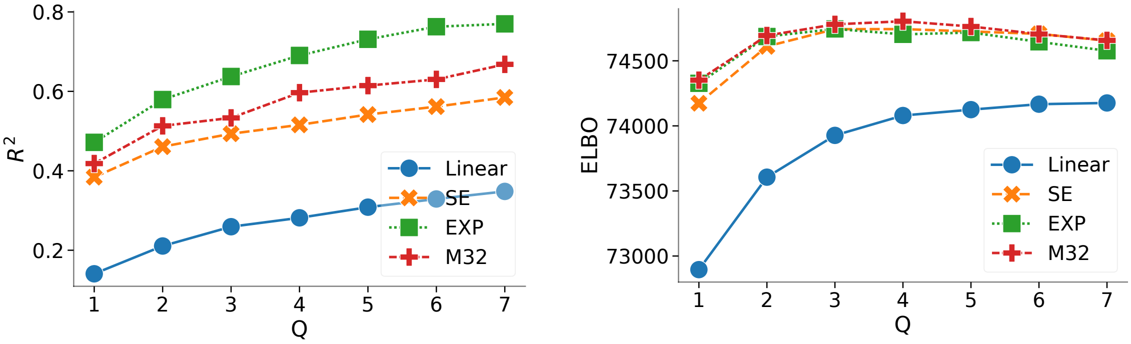


Figure 7.2.3: R^2-score (left) and ELBO (right) as a function of the latent space dimension Q.

the R-squared (R^2) score, which explains the variance of the data picked up by the model. For an observation $\boldsymbol{y} = (y_1, \dots, y_N)^T$ and prediction $\boldsymbol{f} = (f_1, \dots, f_N)^T$, the R^2 is defined as

$$R^2 = 1 - \frac{\sum_{n=1}^{N}(y_n - f_n)^2}{\sum_{n=1}^{N}(y_n - \hat{y})^2} \ , \tag{7.2.3}$$

where $\hat{y} = \sum_n y_n/N$ is the mean of $\boldsymbol{y}$. R^2 equal to one is a perfect fit, in the sense that all the variance was picked up by the model. However, by assuming a lot of noise in the financial data, an R^2 of one is not desired. That would mean, that the noise was also picked up as structure which is the definition of overfitting.

Figure 7.2.3 (left hand side) shows the R^2 score as a function of the latent space dimension Q. Different colors indicate different kernel functions. For every dimension, the nonlinear kernels capture more of the variance than the linear kernel. Within the nonlinear kernels, the exponential kernel (Ornstein-Uhlenbeck process) is the best in terms of the R^2 score. The more dimensions we allow the GPLVM to use, the better the R^2 score gets. However, as mentioned earlier, high R^2 scores are not always desirable, since at some point the model will start learning the noise.

Another measure for the goodness of the fits is the marginal likelihood. The marginal likelihood is used a lot for model selection in Bayesian analysis. However, it is not always computable. We are using variational Bayes to fit the data to the GPLVM. As explained in Section 3.2 the objective function, that we maximize, is the ELBO (Evidence Lower Bound), which is a lower bound to the marginal likelihood (also called the evidence). ELBO can be taken as a proxy to the marginal likelihood. The ELBO also incorporates the complexity (flexibility) of a model by penalizing highly complex models

$$\begin{aligned}\text{ELBO} &= \int q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta}) \log \frac{q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta})}{p(\boldsymbol{B},\boldsymbol{\theta},\boldsymbol{r})} \mathrm{d}\boldsymbol{B}\mathrm{d}\boldsymbol{\theta} \\ &= \int q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta}) \log p(\boldsymbol{r}|\boldsymbol{B},\boldsymbol{\theta}) \mathrm{d}\boldsymbol{B}\mathrm{d}\boldsymbol{\theta} \\ &\quad - \int q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta}) \log \frac{p(\boldsymbol{B},\boldsymbol{\theta})}{q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta})} \mathrm{d}\boldsymbol{B}\mathrm{d}\boldsymbol{\theta} \\ &= \mathbb{E}_{q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta})}[\log p(\boldsymbol{r}|\boldsymbol{B},\boldsymbol{\theta})] - \text{KL}[q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta}) \| p(\boldsymbol{B},\boldsymbol{\theta})] \ . \end{aligned} \tag{7.2.4}$$

The first term in the sum is the data fit term (expectation of the data likelihood under q) and the second term is a regularization, that forces the variational approximation q not to deviate much from the prior. The preference of the variational approximation $q_{\boldsymbol{\nu}}(\boldsymbol{B},\boldsymbol{\theta})$ towards more complex models, such that the data fit term is maximized, is counteracted by the the KL-term.

Figure 7.2.3 (right hand side) shows the ELBO as a function of the latent dimension Q. Here we see, that the model does not need too many dimensions to fit the data. Three or four latent space dimensions are already good enough for the stationary kernel functions. Note that the quantities shown in Figure 7.2.3 are averaged over all the stocks. However, there are differences in the variability picked up by the model from stock to stock. Some stocks are explained better by the model, others not so good.

7.3 Experiments

In this section we are going to apply the GPLVM to real world problems. The GPLVM provides the estimated covariance matrix $\boldsymbol{K}$ of the N stocks and their latent space positions $\boldsymbol{X}$. We use those to construct minimal risk portfolios using the modern portfolio theory (Section 6.2), to predict returns for stocks, which are not traded at a particular day and to match patterns by visualizing the latent space.

7.3.1 Portfolio allocation

The aim of portfolio allocation is to build a portfolio that obey certain criteria. E.g.: Given a number of stocks and their performance in the past, which one of them should we buy for the future to maximize our returns and minimize our risk? Modern portfolio theory (Section 6.2) provides an answer to that question. To optimize a portfolio for risk and return, we need the mean return $\hat{\boldsymbol{r}}$ and the covariance matrix $\boldsymbol{K}$. Then we get the optimal weights $\boldsymbol{w}_{\text{opt}}$ for the portfolio by minimizing Equation (6.2.1). Since we only estimate the covariance matrix $\boldsymbol{K}$ and not the mean returns $\hat{\boldsymbol{r}}$, we build the minimal risk portfolio instead. This is done by minimizing Equation (6.2.2). The minimization is done under the constraint $\sum_n w_n = 1$ and $0 < w_n < 0.1$. The first constraint ensures that we are fully invested, $w_n > 0$ ensures that we go long only (we only buy stocks and do not short sell them) and the second constraint ($w_n < 0.1$) ensures that we do not put too much weight on a single asset.

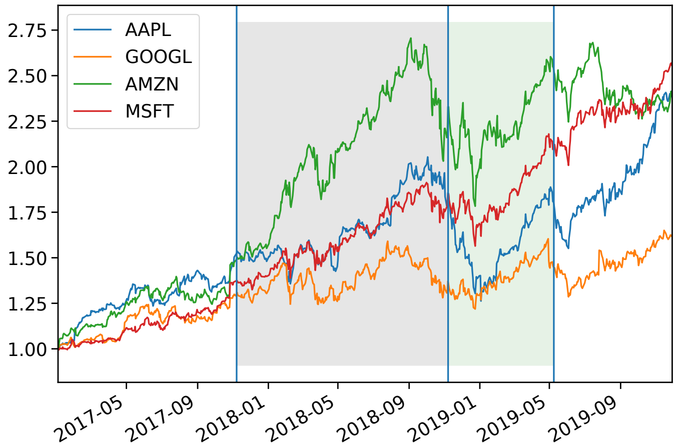


Figure 7.3.1: Learning period of one year is highlighted in light grey and prediction/holding period of six months is highlighted in light green.

The procedure is the following: We select 60 stocks at random from the S&P500 from Jan 2008 to Jan 2018. Then we learn $\boldsymbol{w}_{\text{opt}}$ from the past year and buy accordingly for the next six months. We start from Jan 2008 (learning $\boldsymbol{w}_{\text{opt}}$ from the data from Jan 2008 till Dec 2008 and by stocks according to $\boldsymbol{w}_{\text{opt}}$ for Jan 2009 till June 2009) and repeat the procedure every six months. The learning and holding period is highlighted in Figure 7.3.1. The learning period is highlighted in light gray and the holding period is shown in light green. Note that we are not learning from the price as shown in the Figure but from the returns for the same period. By "learning" we mean the estimation of the covariance matrix $\boldsymbol{K}$ by the GPLVM. Based on the kernel function we choose, we get different results. We repeat the experiment for the same stocks for different kernels and also include the performance for the sample covariance (i.e.: $\boldsymbol{K} = \frac{1}{D}(\boldsymbol{r} - \hat{\boldsymbol{\mu}})(\boldsymbol{r} - \hat{\boldsymbol{\mu}})^T$, where $\hat{\boldsymbol{\mu}}_n = \frac{1}{D}\sum_{d=1}^{D} \boldsymbol{r}_{nd}$) and the shrunk Ledoit-Wolf covariance matrix (Ledoit and Wolf, 2004)[4]. We also included the equally weighted portfolio, where $\boldsymbol{w}_{\text{opt}} = (1, \ldots, 1)/N$ [5]. Both estimators (Ledoit-Wolf and sample covariance) are explained in Section 6.3. The performance of the portfolios of different models is evaluated using the mean return, standard deviation and the Sharpe ratio on a yearly basis. Sharpe ratio (Sharpe, 1966) is a measure for the performance of a portfolio, that includes both, the risk and return of that portfolio. It is defined as the average return earned per unit of volatility and can be calculated by dividing the mean of a series of returns by its standard deviation.

Table 7.2 shows the performance for different models. Nonlinear kernels have the minimal variance and at the same time the best Sharpe ratio values. Note, however, that we are not maximizing the mean, since we do not have an estimator for that. For a finite q in

[4] We have used the implementation in the Python toolbox scikit-learn (Pedregosa et al., 2011) in the experiments.

[5] Estimating historical returns and high-dimensional covariance matrices is very challenging. Jobson and Korkie (1981) show that often times an equally weighted portfolio outperforms the portfolio constructed from sample estimates.

Table 7.2: Mean return, standard deviation and the Sharpe ratio of different models on a yearly basis (Nirwan and Bertschinger, 2019b).

MODEL	LINEAR	SE	EXP	M32	SAMPLE COV	LEDOIT W	EQUALY W
MEAN	0.142	0.151	0.155	0.158	0.149	0.148	0.182
STD	0.158	0.156	0.154	0.153	0.159	0.159	0.232
SHARPE RATIO	0.901	0.969	1.008	1.029	0.934	0.931	0.786

Equation (6.2.1) one can also build portfolios, which not only minimize volatility but also maximize the returns. This, however, is not the goal of our research.

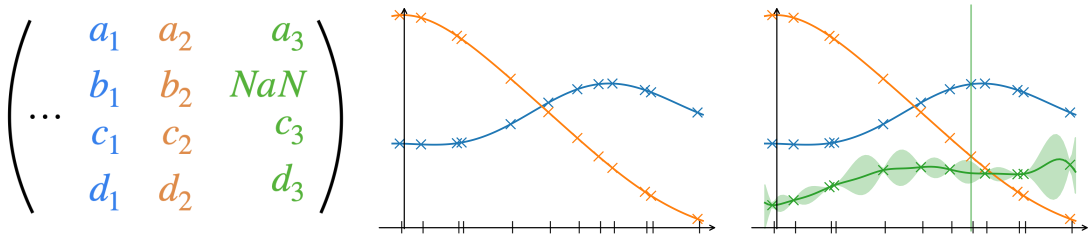
7.3.2 Fill in missing values

Regulation requires fair value assessment of all assets (Financial Accounting Standards Board, 2006), including illiquid and infrequently traded ones. Liquidity refers to the ease with which an asset can be sold and the price of the asset at a particular moment is the price of the last sale. The assessment of a portfolio (or just one asset) needs the price of all the assets inside that portfolio. If, however, an asset is not traded on a particular day, there will be no price for that asset at that day. An approximation to that missing value is, for example, the price at which the asset was traded last time or the average over last few trades. These, however, are not good estimators of the price.

We demonstrate how the GPLVM can be used to fill in missing prices/returns. The procedure is the following: We take the data matrix without the missing values and learn the latent space positions $\boldsymbol{B}$ for all the assets and the kernel hyperparameters $\boldsymbol{\theta} = (\boldsymbol{\sigma}, l, \boldsymbol{\sigma}_{\text{noise}})$. Given the positions and the hyperparameters, we can now train a standard Gaussian process (GP) (Equation 4.2.8) and predict the return of the illiquid asset given the return of all the other assets at a particular day. The procedure is illustrated in Figure 7.3.2 where in subfigure (a) the blue and the orange values are the days (there might be many more), where the return of the asset that we want to predict, was observed. The green column represents a day, where the return was not observed (Figure 7.3.2 (a)). We can now learn the latent space with the blue and the orange column (Figure 7.3.2 (b)) and train a standard GP on the green column to get the predictive distribution for the missing return (Figure 7.3.2 (c)). This toy example also shows, that the predictive density conditioned on the values of other assets at the same day is a much better predictor than the value of the asset a day ago or an average over the past few days[6].

For illustration purposes, we continue working with the prices of stocks from the S&P500. The return matrix $\boldsymbol{r}$ is first split into a training and a test set. In the test data set we discard the return of an asset and the goal is to use the GP and GPLVM to predict the return for that asset. Given N stocks on day d, we fit a GP to $N-1$ stocks and predict

[6] Remember, that the return values are more or less independent over days, thus do not contain any (or not enough) information for the value on successor days.



(a) return matrix $\boldsymbol{r}$ with missing value.

(b) learning latent space positions and hyperparameters.

(c) training a GP to predict missing values.

Figure 7.3.2: Illustration of the procedure for predicting missing values.

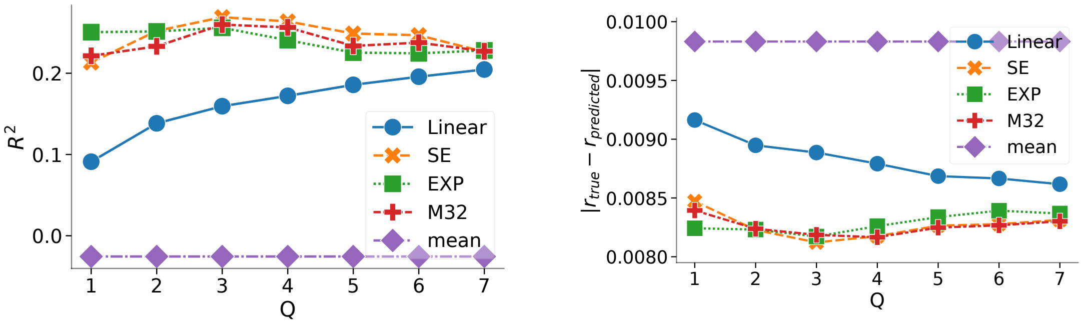


Figure 7.3.3: Left: R^2 score of the predicted values. Right: Average absolute deviation of the predicted return to the real return evaluated by Leaving-one-out cross-validation.

the return of the remaining stock. We iterate thorough all stock for D_{test} days, such that every stock return is predicted once for each of the D_{test} days.

Figure 7.3.3 shows the R^2-score (left) and the average absolute deviation $\frac{1}{ND_{\text{test}}}\sum_{nd}|\boldsymbol{r}_{nd} - \boldsymbol{r}_{nd}^{\text{pred}}|$ as function of the latent space dimension Q for different models. We see that already a linear model is better than the mean of the historic values. Note that the R^2-score even gets negative. The nonlinear kernels perform best. The same is true for the absolute deviation from the true value (Figure 7.3.3 right hand side). Here, the nonlinear kernels perform best for $Q = 3$.

7.3.3 Interpretation of the latent space

Even though the ELBO was the highest for Q between three and five, it still helps to visualize the latent space for $Q = 1$ and $Q = 2$, to get a better understanding of the model.

Equation (7.1.1) fits the daily returns given the latent space positions $\boldsymbol{B}$. The distribution over this function is dependent on the kernel function. For the linear kernel the samples from the posterior distribution are linear functions, for the SE they are not linear anymore

and are infinitely often differentiable and for the exponential kernel they are nonlinear as well but they are not differentiable at all. To get an intuition for the fit, in Figure 7.3.4 we show the mean of the posterior GP for two random days for the three kernel functions.[7]
Here, we see how the model tries to capture the structure. It places the stocks according to their correlation somewhere on the abscissa and then tries to fit them with a function. The form of this function is given by the choice of the kernel. The linear model tries to fit the points by a linear function and has to explain the rest as noise. It is obvious, that if we allow nonlinear functions, we can capture more structure and thereby reducing the part stated as noise. Also note that there is still a lot of variability in the data, that is not captured by the model. The model is able to capture more variability when we allow for more than just one latent space dimension.

The same happens in two and more dimensions but we cannot visualize that anymore. The 2-d latent space can be visualized using a scatter plot. Figure 7.3.5 shows that latent space for the RBF kernel. Stationary kernels are distance dependent and the distance is directly related to the correlation of the stocks. If the distance between two stocks is low, the correlation will be high. That means, that stocks should cluster according to their correlation. That is exactly what we see in Figure 7.3.5. Stocks from the same sector are marked by different marker style and color. As we see, they tend to cluster together. We consider our method as an alternative to other methods for detecting structure in financial data.

[7] This plot is made by standardizing the return matrix $\boldsymbol{r}$ and setting $\boldsymbol{\sigma} = \sigma(1, \ldots, 1)^T$, such that the scale of returns of all assets becomes the same.

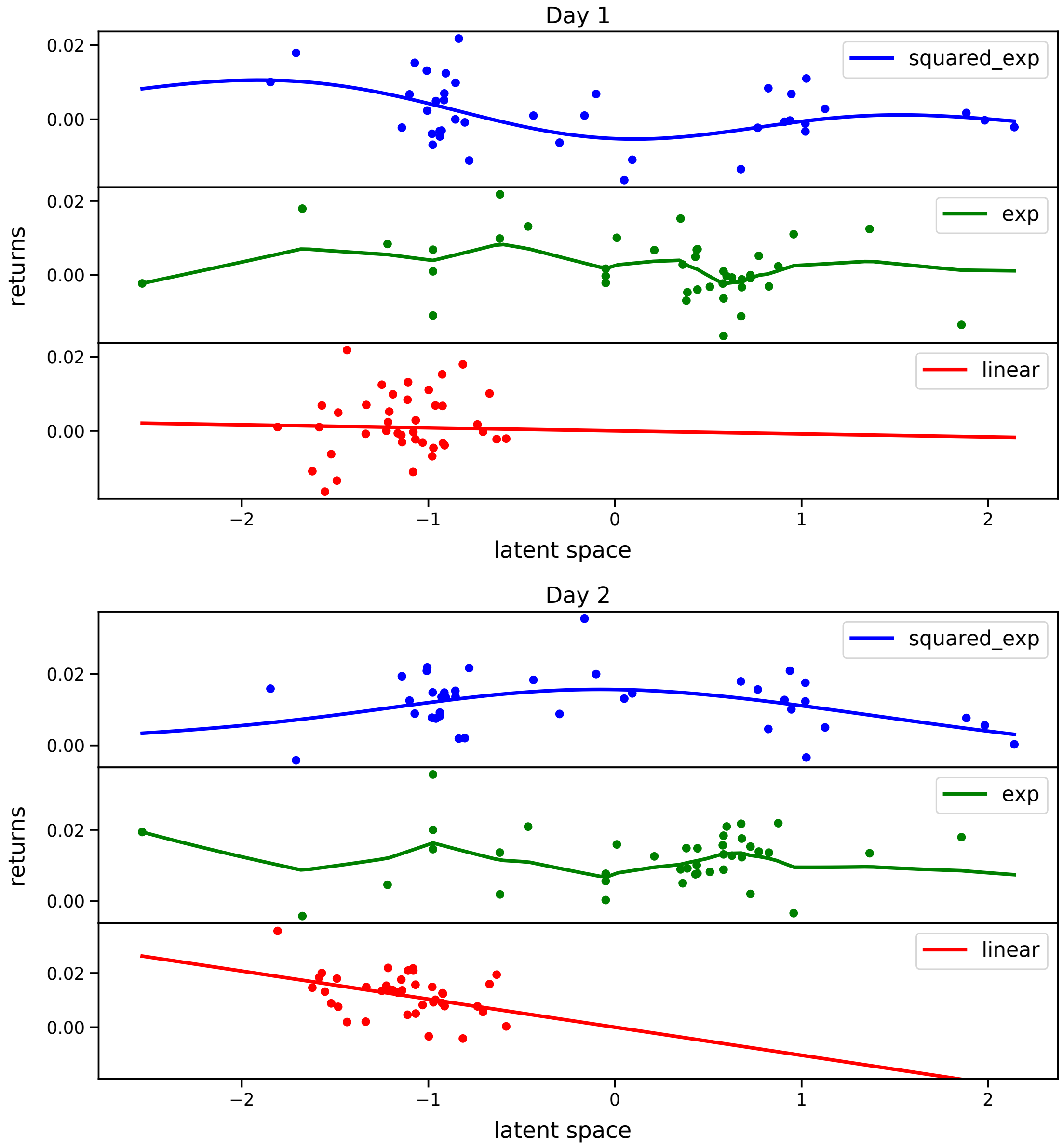


Figure 7.3.4: Mean function of the posterior GP for 50 stocks.

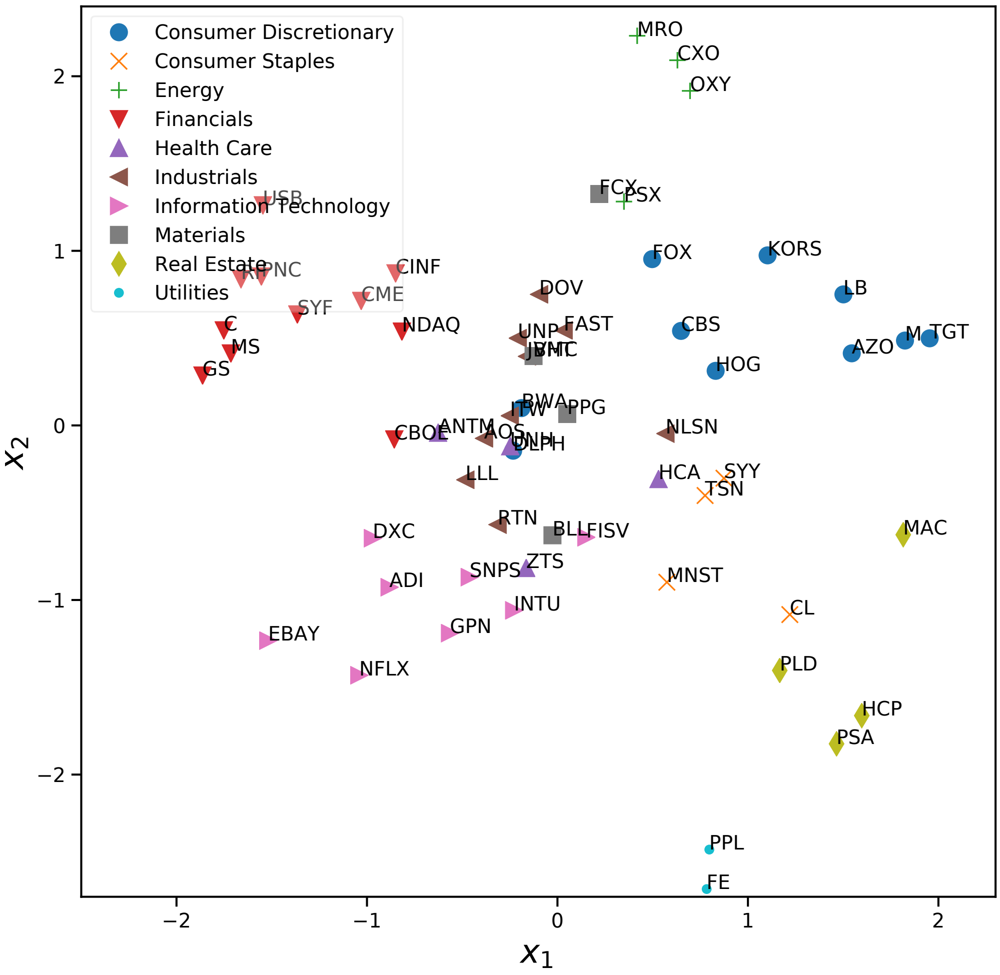


Figure 7.3.5: Scatter plot of the 2-d latent space for the RBF-kernel.

Chapter 8

Rotation invariant Householder parameterization for Bayesian PCA

We discussed the probabilistic PCA and the GPLVM in Chapter 5 and used the GPLVM for modeling financial data in Chapter 7. While doing that we encountered rotational symmetry in the latent space in PPCA as well as in GPLVM. That symmetry only appears in the posterior of the latent positions $\boldsymbol{X}$ or $\boldsymbol{B}$ and does not affect the data distribution $p(\boldsymbol{Y})$ nor the predictive distribution $p(\boldsymbol{y}'|\boldsymbol{Y})$. Howerver, it causes problems, when interpreting the latent space[1]. In this chapter we propose a new parametrization based on Householder transformations, which removes the rotational symmetry of the posterior, without changing $p(\boldsymbol{Y})$ and $p(\boldsymbol{y}'|\boldsymbol{Y})$, i.e. the model.
The work in this section is published at the International Conference on Machine Learning 2019 (Nirwan and Bertschinger, 2019a).

8.1 Problem definition

The marginal likelihood of PPCA has the form

$$p(\boldsymbol{Y}|\boldsymbol{W}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{y}_n|\boldsymbol{0}, \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}) \ , \tag{8.1.1}$$

as we derived in Chapter 5. Since it only depends on the outer product of $\boldsymbol{W}$, a rotation of the matrix will result in the same likelihood, i.e.: $p(\boldsymbol{Y}|\boldsymbol{W}) = p(\boldsymbol{Y}|\boldsymbol{W}\boldsymbol{R}) \ \forall \ \boldsymbol{R} \in \{\boldsymbol{R} \mid \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}\}$, since $\boldsymbol{W}\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^T = \boldsymbol{W}\boldsymbol{W}^T$. This rotational symmetry is illustrated

[1] We do not see that rotation invariance in Figure 7.3.5 (latent space visualization of the stocks) since the figure only shows the mean of the variational distribution. Also our variational approximation approximates the posterior by independent Gaussians, which are not able to capture the rotation. However, if we were to fit the model using variational inference more than ones, the solutions would differ only by a rotation (excluding differences due to different local minima).

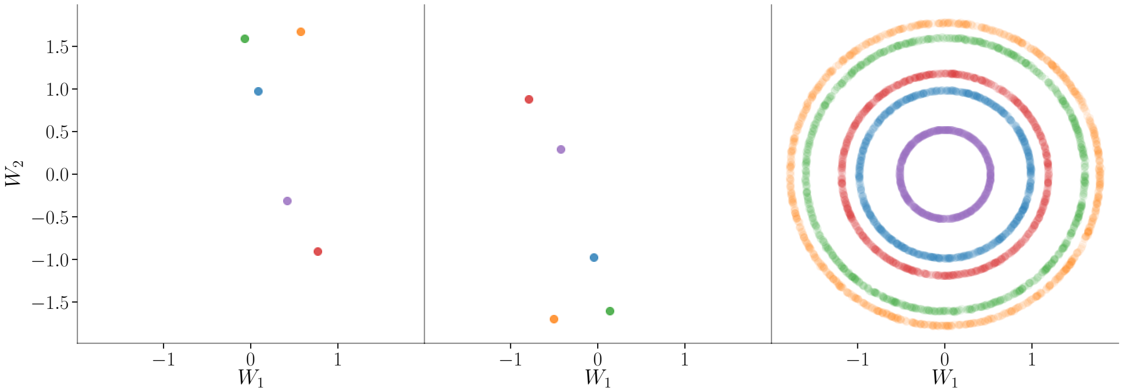


Figure 8.1.1: Illustration of the rotational symmetric maximum likelihood solution.

for a 2-dimensional latent space in Figure 8.1.1, where the first two subplots show the maximum likelihood solution for two different initial conditions. Each row of $\boldsymbol{W}$ is shown different colors. The resulting matrices for both runs are similar in a sense that the likelihood is the same for both of them. This is the case because both results only differ by a rotation, which does not affect the likelihood. The right subfigure shows the maximum likelihood solution for 1000 different initial conditions. Here, we clearly see the rotational symmetry.

For a Bayesian analysis of the model, we assume a prior on $\boldsymbol{W}$ and infere the posterior $p(\boldsymbol{W}|\boldsymbol{Y})$ using Bayes rule

$$p(\boldsymbol{W}|\boldsymbol{Y}) = \frac{p(\boldsymbol{Y}|\boldsymbol{W})p(\boldsymbol{W})}{p(\boldsymbol{Y})} \,. \tag{8.1.2}$$

Most of the time, a Gaussian prior is chosen for $\boldsymbol{W}$. A Gaussian prior is rotationally symmetric and does not choose an a-priori direction in the latent space and the data space. That is desired if we do not know the directions of both spaces and also do not want to prefer any rotation a-priori. However, since the likelihood $p(\boldsymbol{Y}|\boldsymbol{W})$ is rotational symmetric as well, the posterior will also have the same symmetry. Draws from the posterior are shown in Figure 8.1.2, where we used HMC to draws 4000 samples. Again, it exhibits the rotational symmetry.

8.2 Background

To break the rotational symmetry without changing the model, we will follow a similar ansatz to Tipping and Bishop (1999). We decompose $\boldsymbol{W}$ into $\boldsymbol{U\Sigma V}^T$ using singular value decomposition. A latent space point $\boldsymbol{x} \in \mathbb{R}^Q$ is now mapped to the data space by first a rotation $\boldsymbol{V} \in \mathbb{R}^{Q\times Q}$, followed by a scaling via the diagonal matrix $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \ldots, \sigma_Q) \in \mathbb{R}^{Q\times Q}$ and then by a rotation via $\boldsymbol{U} \in \mathbb{R}^{D\times Q}$. Since the directions in the latent space are arbitrary, due to the symmetry, $\boldsymbol{V}$ is not uniquely identified. Luckily, the outer product

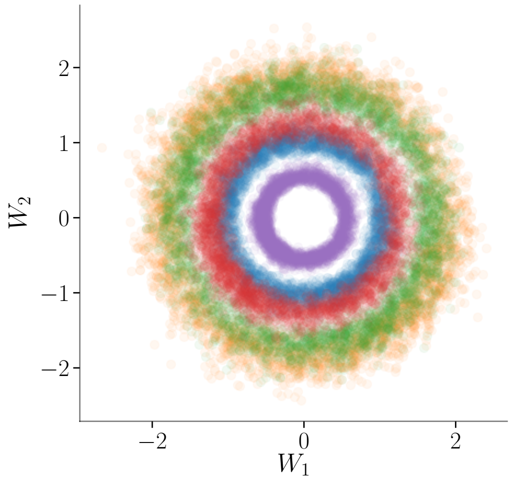


Figure 8.1.2: Illustration of the rotational symmetric posterior $p(\boldsymbol{W}|\boldsymbol{Y})$.

does not depend on $\boldsymbol{V}$, since

$$\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{\Sigma}^T\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\boldsymbol{U}^T \ . \tag{8.2.1}$$

Therefore, the choice of $\boldsymbol{V}$ does not change the model. We set $\boldsymbol{V} = \boldsymbol{I}$, by which we fix the unidentifiability of the latent space.
However, since we reparameterized the model, we need to find the proper priors for the new parameterization, such that the model stays unchanged. For that, we need concepts from random matrix theory. This section introduces those concepts.

8.2.1 Singular Value Decomposition

Singular value decomposition decomposes any matrix $\boldsymbol{W} \in \mathbb{R}^{D\times Q}$ as

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \ , \tag{8.2.2}$$

where $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D)^T \in \mathbb{R}^{D\times Q}$ and $\boldsymbol{V} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_D)^T \in \mathbb{R}^{Q\times Q}$ are orthogonal matrices and $\boldsymbol{\Sigma} = \text{diag}(\sigma, \ldots, \sigma_Q) \in \mathbb{R}^{D\times Q}$ is a diagonal matrix containing the singular values. The rank P of $\boldsymbol{W}$ ($P \leq Q$) is the number of non-zero singular values, which are the eigenvalues of $\boldsymbol{W}\boldsymbol{W}^T$ and $\boldsymbol{W}^T\boldsymbol{W}$. $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ obey the relation

$$\begin{aligned} &\boldsymbol{W}\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i, \quad &&\boldsymbol{W}^T\boldsymbol{u}_i = \sigma_i\boldsymbol{v}_i, \quad &&\forall\ i = 1, \ldots, P \ , \\ &\boldsymbol{W}\boldsymbol{v}_i = \boldsymbol{0}, &&\boldsymbol{W}^T\boldsymbol{u}_i = \boldsymbol{0}, &&\forall\ i > P \ . \end{aligned} \tag{8.2.3}$$

The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are called the left-singular eigenvectors and right-singular eigenvectors of $\boldsymbol{W}$, respectively.

8.2.2 Random Matrix Theory

As mentioned in previous chapters, the prior on $\boldsymbol{W}$ is chosen to be a standard Gaussian most of the time in PPCA. A Gaussian prior preserves the rotation invariance in the data space. That is preferable when we do not know the principle directions in the data space a-priori. However, this choice of the prior also preserves the latent space rotation, which leads to unidentified latent space directions. We fix this by using the SVD on the mapping $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and set $\boldsymbol{V} = \boldsymbol{I}$. In this subsection, we investigate how to choose the prior on $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ such that we do not change the marginal likelihood (i.e. the model). Specifically, we look for $p(\boldsymbol{U}, \boldsymbol{\Sigma})$ such that

$$\int p(\boldsymbol{Y}|\boldsymbol{W})p(\boldsymbol{W})\mathrm{d}\boldsymbol{W} = \int p(\boldsymbol{Y}|\boldsymbol{U}, \boldsymbol{\Sigma})p(\boldsymbol{U}, \boldsymbol{\Sigma})\mathrm{d}\boldsymbol{U}\mathrm{d}\boldsymbol{\Sigma}\,. \tag{8.2.4}$$

Therefore, we use results from the random matrix theory. It turns out that the data space rotation $\boldsymbol{U}$ and the singular values $\boldsymbol{\Sigma}$ are independent for a product of a Gaussian matrix (which is a Wishart matrix) $p(\boldsymbol{U}, \boldsymbol{\Sigma}) = p(\boldsymbol{U})p(\boldsymbol{\Sigma})$. The independence of $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ and their exact distribution follow from the theorems described below.

Prior on the singular values $\boldsymbol{\Sigma}$

For a standard Gaussian prior on $\boldsymbol{W}$, $\boldsymbol{W}\boldsymbol{W}^T$ has a Wishart distribution and the following theorem holds (James and Lee, 2014)

Theorem 8.2.1 *Let the entries of $\boldsymbol{W} \in \mathbb{R}^{D\times Q}$ be i.i.d. Gaussian with zero mean and unit variance. The joint probability density of the ordered strictly positive eigenvalues of the Wishart matrix $\boldsymbol{W}^T\boldsymbol{W}$, $\lambda_1 \geq ... \geq \lambda_Q$, equals*

$$p(\boldsymbol{\lambda}) = ce^{-\frac{1}{2}\sum_{q=1}^{Q}\lambda_q}\prod_{q=1}^{Q}\left(\lambda_q^{\frac{D-Q-1}{2}}\prod_{q'=q+1}^{Q}|\lambda_q - \lambda_{q'}|\right)\,, \tag{8.2.5}$$

where $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_Q)$ and c is a constant that depends on D and Q.

The non-zero eigenvalues of $\boldsymbol{W}^T\boldsymbol{W}$ are the same as that of $\boldsymbol{W}\boldsymbol{W}^T$. Therefore they have the same probability density function. The singular values are given by the square roots of the variance $\boldsymbol{\lambda}$, i.e. $\sigma_i = \sqrt{\lambda_i}$. Thus, we get

$$p(\sigma_1, ..., \sigma_Q) = ce^{-\frac{1}{2}\sum_{q=1}^{Q}\sigma_q^2}\prod_{q=1}^{Q}\left(\sigma_q^{D-Q-1}\prod_{q'=q+1}^{Q}|\sigma_q^2 - \sigma_{q'}^2|\right)\prod_{q=1}^{Q}2\sigma_q\,, \tag{8.2.6}$$

where the last product in the term above is the Jacobian correction

$$p_\sigma(\boldsymbol{\sigma}) = p_\lambda(g^{-1}(\boldsymbol{\sigma}))\left\|\det\left(\frac{\mathrm{d}g^{-1}(\boldsymbol{\sigma})}{\mathrm{d}\boldsymbol{\sigma}}\right)\right\| = p_\lambda(g^{-1}(\boldsymbol{\sigma}))\prod_{q=1}^{Q}2\sigma_q\,, \tag{8.2.7}$$

with $\boldsymbol{\lambda} = g^{-1}(\boldsymbol{\sigma}) = (\sigma_1^2, \ldots, \sigma_Q^2)$. Equation (8.2.7) is the right prior distribution for the singular values $\boldsymbol{\Sigma}$. Note that the singular values are ordered and the prior for $\boldsymbol{\Sigma}$ does not depend on $\boldsymbol{U}$.

Prior on the principle components $\boldsymbol{U}$

As mentioned earlier, a standard Gaussian prior on $\boldsymbol{W}$ results into a Wishart distribution on $\boldsymbol{W}\boldsymbol{W}^T$. A Wishart matrix can be decomposed into a product of an orthogonal matrix $\boldsymbol{U}$ and a scaling matrix $\boldsymbol{\Sigma}$ as show in Equation (8.2.1). $\boldsymbol{U}$ is orthogonal by construction (see singular value decomposition). The set of D times Q orthogonal matrices is called the Stiefel manifold $\mathcal{V}_{Q,D}$

$$\mathcal{V}_{Q,D} = \left\{ \boldsymbol{U} \in \mathbb{R}^{D \times Q} | \boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I} \right\} \; . \tag{8.2.8}$$

So, $\boldsymbol{U}$ is from the Stiefel manifold. Now we just need the right measure on the Stiefel manifold. It is known that the eigenvectors of a Wishart matrix are distributed uniformly in the space of orthogonal matrices[2]. The eigenvector matrix of a Wishart distributed matrix, as well as of a singular Wishart distributed matrix, is Haar-distributed (Bai et al., 2007; Uhlig, 1994)[3]. The dimension of the Stiefel manifold (D times Q orthogonal matrices) is $DQ - \frac{1}{2}Q(Q+1)$, accounting for the fact that the orthogonality constraint reduces the number of independent degrees of freedom.
Now we know that for a Wishart matrix the singular values are distributed according to Equation (8.2.6) and the principle components are uniformly distributed on the Stiefel manifold. We can directly sample from the probability density function of $\boldsymbol{\Sigma}$, which is already known. Though we know that the density of $\boldsymbol{U}$ on the Stiefel manifold (space of orthogonal matrices) is uniform (Haar-distributed), we still need to parameterize that manifold and sample the parameters. We need to find an unconstrained parameterization for orthogonal matrices along with a density on the parameters, such that the resulting matrix is Haar distributed. For that, we will use the Householder parameterization as explained in the next section.

8.2.3 Householder transformations

Pourzanjani et al. (2017); Shepard et al. (2014) and Mezzadri (2007) proposed different approaches to sample uniformly from the Stiefel manifold. Pourzanjani et al. (2017) parameterize the space using givens rotations, which requires the calculation of the Jacobian correction for the parameters since the exact distribution of the parameters is unknown. The Jacobian correction, however, is computationally very demanding. We follow the approach of Shepard et al. (2014) and Mezzadri (2007) and parameterize the Stiefel manifold using Householder reflections. In addition, Mezzadri (2007) provides the right density on the parameters such that the resulting orthogonal matrix has a uniform distribution on the Stiefel manifold. As a consequence, we do not need to calculate the Jacobian correction for the change of measure. As we will see, this does not produce a computational overhead and the complexity of our algorithm is the same as that of PPCA. This section discusses the theory behind the procedure, with some visual aids.

[2] Note that $\boldsymbol{W}$ is only of rank Q. Thus $\boldsymbol{W}\boldsymbol{W}^T \in \mathbb{R}^{D \times D}$ is a singular matrix.

[3] The uniform measure on the Stiefel manifold is the Haar-measure. The cited papers claim that the distribution of singular Wishart matrices are also uniform. We could not find any proof for that claim.

QR-decomposition

Given a matrix $\boldsymbol{Z} \in \mathbb{R}^{D\times Q}$, the thin QR-decomposition decomposes $\boldsymbol{Z}$ into a product of an orthogonal matrix $\boldsymbol{Q} \in \mathcal{V}_{Q,D}$ from the Stiefel manifold and an upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{Q\times Q}$, $\boldsymbol{Z} = \boldsymbol{Q}\boldsymbol{R}$. If the elements of $\boldsymbol{Z}$ are i.i.d. standard Gaussian, $\boldsymbol{Q}$ will be Haar distributed (Mezzadri, 2007). However, this only holds when a unique QR-decomposition is applied. A unique decomposition is achieved by enforcing the convention that the diagonal elements of $\boldsymbol{R}$ are positive.

One way to compute the QR-decomposition of $\boldsymbol{Z}$ is by using the Householder transformations $\boldsymbol{H}_d$. These are reflections on the plane spanned by a vector $\boldsymbol{u}_d \in \mathbb{R}^d$. The first reflection $\boldsymbol{H}_D$ is chosen in a way that the application of $\boldsymbol{H}_D$ on $\boldsymbol{Z}$ turns its first column to a vector aligned with the first coordinate axis

$$\boldsymbol{H}_D\boldsymbol{Z} = \begin{pmatrix} \alpha_1 & * & \dots & * \\ 0 & & & \\ \vdots & & \boldsymbol{Z}' & \\ 0 & & & \end{pmatrix} . \tag{8.2.9}$$

We need to reflect the first column $\boldsymbol{z}_1$ of $\boldsymbol{Z} = (\boldsymbol{z}_1, \dots, \boldsymbol{z}_D)^T$ in such a way that all coordinates but one disappear. The choice of the plane with the normal $\hat{\boldsymbol{u}}_D$ corresponding to the transformation $\boldsymbol{H}_D$, that achieves exactly that, is illustrated in Figure 8.2.1 and is given by

$$\begin{aligned} \boldsymbol{H}_D &= \boldsymbol{I} - 2\hat{\boldsymbol{u}}_D\hat{\boldsymbol{u}}_D^T \\ \hat{\boldsymbol{u}}_D &= \frac{\boldsymbol{z}_1 \pm \|\boldsymbol{z}_1\|\boldsymbol{e}_1}{\left\|\boldsymbol{z}_1 \pm \|\boldsymbol{z}_1\|\boldsymbol{e}_1\right\|} , \end{aligned} \tag{8.2.10}$$

where $\boldsymbol{z}_1$ is the first column of $\boldsymbol{Z}$ and $\boldsymbol{e}_1 = (1, 0, 0, \dots)^T$. Successive applications of Householder transformations $\boldsymbol{H}'_d \in \mathbb{R}^{d\times d}$ on the submatrices $\boldsymbol{Z}' \in \mathbb{R}^{d\times d}$ will lead to an upper triangular matrix $\boldsymbol{R} = \prod_d \boldsymbol{H}_d\boldsymbol{Z}$, where

$$\boldsymbol{H}_d = \begin{pmatrix} \boldsymbol{I}_{d-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}'_d \end{pmatrix} . \tag{8.2.11}$$

The product of the Householder transformations $\prod_d \boldsymbol{H}_d$ is the orthogonal matrix $\boldsymbol{Q}$.

Thus, to sample uniformly from the Stiefel manifold, we could simply QR-decompose a matrix $\boldsymbol{Z}$ with each element being drawn from a standard Gaussian, $\boldsymbol{Z}_{ij} = \mathcal{N}(0, 1)$. However, this would totally overparameterize the model, since $\boldsymbol{Z}$ has $D * Q$ degrees of freedom.

Note that for the procedure above, the relevant parts for $\boldsymbol{Q} \in \mathcal{V}_{Q,D}$ are the Householder transformations $\boldsymbol{H}'_d \in \mathbb{R}^{d\times d}$, which only require the normal vector $\hat{\boldsymbol{u}}_d \in \mathbb{R}^d \ \forall\ d \in 1, \dots, D$. This vector, for every d, is created by $\boldsymbol{z}'_d \in \mathbb{R}^d$, where each $\boldsymbol{z}'_d$ is the first column of the submatrix $\boldsymbol{Z}'$ and has far less parameters than $\boldsymbol{Z}$. Thus, we can find a more compact parameterization by using these insights.

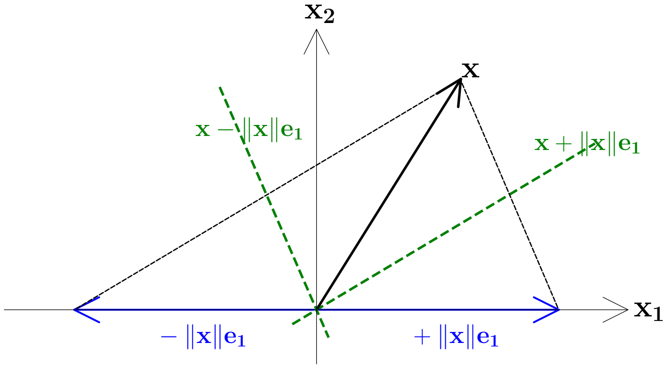

$\mathbf{x}_2$
$\mathbf{x}$
$\mathbf{x} - \|\mathbf{x}\|\mathbf{e}_1$
$\mathbf{x} + \|\mathbf{x}\|\mathbf{e}_1$
$\mathbf{x}_1$
$-\|\mathbf{x}\|\mathbf{e}_1$
$+\|\mathbf{x}\|\mathbf{e}_1$

Figure 8.2.1: Householder transformation, which maps the vector $\boldsymbol{x}$ to $\pm\|\boldsymbol{x}\|\boldsymbol{e}_1$.

Algorithm to sample uniformly from Stiefel manifold

Mezzadri (2007) provides an efficient way to sample an orthogonal matrix $\boldsymbol{U} \in \mathbb{R}^{D\times Q}$ according to the Haar measure. We sample $\boldsymbol{v} = (\boldsymbol{v}_D, \boldsymbol{v}_{D-1}, \ldots, \boldsymbol{v}_{D-Q+1})$ from $p(\boldsymbol{v})$ (see Theorem 8.2.2), where $\boldsymbol{v}_d \in \mathbb{R}^d \ \forall\ d \in \{D-Q+1, \ldots, D\}$ and transform them via a particular form of the Householder transformations to $\boldsymbol{Q}$. This particular form ensures the requirement of a unique QR-decomposition (positive diagonal of $\boldsymbol{R}$). $\boldsymbol{U}$ is the product of Q Householder transformations

$$\boldsymbol{U} = \boldsymbol{H}_D(\boldsymbol{v}_D)\boldsymbol{H}_D - 1(\boldsymbol{v}_{D-1}) \ldots \boldsymbol{H}_{D-Q+1}(\boldsymbol{v}_{D-Q+1}) \ . \tag{8.2.12}$$

To construct $\boldsymbol{H}_d$, we define $\tilde{\boldsymbol{H}}_d(\boldsymbol{v}_d) \in \mathbb{R}^{d\times d}$ as

$$\tilde{\boldsymbol{H}}_d(\boldsymbol{v}_d) = -\mathrm{sgn}(\boldsymbol{v}_{d1})(\boldsymbol{I} - 2\boldsymbol{u}_d\boldsymbol{u}_d^T) \ , \tag{8.2.13}$$

where

$$\boldsymbol{u}_d = \frac{\boldsymbol{v}_d + \mathrm{sgn}(\boldsymbol{v}_{d1})\|\boldsymbol{v}_d\|\boldsymbol{e}_1}{\|\boldsymbol{v}_d + \mathrm{sgn}(\boldsymbol{v}_{d1})\|\boldsymbol{v}_d\|\boldsymbol{e}_1\|} \ , \tag{8.2.14}$$

and construct $\boldsymbol{H}_d$ by

$$\boldsymbol{H}_d = \begin{pmatrix} \boldsymbol{I}_{d-1} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{H}}_d \end{pmatrix} \ . \tag{8.2.15}$$

The choice of the sign in Equation (8.2.13) and Equation (8.2.14) fulfills the requirement of the uniqueness. The following theorem shows which distribution $p(\boldsymbol{v})$ the $\boldsymbol{v}$ need to follow in order to get an orthogonal matrix distributed according to the Haar-measure (Mezzadri, 2007)

Theorem 8.2.2 *Let $\boldsymbol{v}_D, \boldsymbol{v}_{D-1}, ..., \boldsymbol{v}_1$ be uniformly distributed on the unit spheres $\mathbb{S}^{D-1}, ..., \mathbb{S}^0$ respectively, where $\mathbb{S}^{n-1}$ is the unit sphere in $\mathbb{R}^n$. Furthermore, let $\boldsymbol{H}_d(\boldsymbol{v}_d)$ be the d-th Householder transformation as defined in Equation (8.2.13). The product*

$$\boldsymbol{Q} = \boldsymbol{H}_D(\boldsymbol{v}_D)\boldsymbol{H}_{D-1}(\boldsymbol{v}_{D-1})...\boldsymbol{H}_1(\boldsymbol{v}_1) \tag{8.2.16}$$

is a random orthogonal matrix with distribution given by the Haar measure on $O(D)$.

A corresponding draw form the Stiefel manifold $\mathcal{V}_{Q,D}$ can be obtained by only taking the first Q columns of $\boldsymbol{Q}$. Alternatively, as defined in Equation (8.2.12), we only draw $\boldsymbol{v}_D, \dots, \boldsymbol{v}_{D-Q+1}$ vectors from the respective sphere and construct the orthogonal matrix from the corresponding Householder transformations. This works, since the Householder transformations $\boldsymbol{H}_{D-Q}, \dots, \boldsymbol{H}_1$ only effect the columns $D - Q$ to 1 (Equation 8.2.15).

2-d example for a sample from $\mathcal{V}_{2,2}$

Here, we consider an example for a construction of a 2-d orthogonal matrix $\boldsymbol{U}$. We need two vectors $\boldsymbol{v}_1 = (0, v_{11})^T$ and $\boldsymbol{v}_2 = (v_{21}, v_{22})^T$, where $v_{11}, v_{21}, v_{22} \sim \mathcal{N}(0, 1)$. Using Equation (8.2.15), we get

$$\boldsymbol{H}_1 = \begin{pmatrix} 1 & 0 \\ 0 & v_{11} \end{pmatrix} . \tag{8.2.17}$$

The construction of $\boldsymbol{H}_2$ is given by $\boldsymbol{v}_2$ that obey $H_2(\boldsymbol{v}_{21}, \boldsymbol{v}_{22})^T = |v_2|\boldsymbol{e}_1 = \boldsymbol{e}_1$. Note, since $\boldsymbol{H}_2 = \boldsymbol{H}_2^{-1}$, we get $\boldsymbol{H}_2\boldsymbol{e}_1 = \boldsymbol{v}_2$. The orthogonality in the columns of $\boldsymbol{U}$ is build by $\boldsymbol{H}_2$, since $\boldsymbol{U} = \boldsymbol{H}_2\boldsymbol{H}_1$. Thus, we get

$$\boldsymbol{U} = \boldsymbol{H}_2\boldsymbol{H}_1 = \left[\boldsymbol{H}_2 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; \boldsymbol{H}_2 \begin{pmatrix} 0 \\ v_{11} \end{pmatrix}\right] = [\boldsymbol{v}_2 \; \boldsymbol{H}_2\boldsymbol{v}_1] . \tag{8.2.18}$$

Therefore, $\boldsymbol{H}_2$ is the matrix, that maps $\boldsymbol{e}_1$ to $\boldsymbol{v}_2$ and "lifts up" v_{11} to 2 dimensions, such that $\boldsymbol{H}_2(0, v_{11})^T \perp \boldsymbol{v}_2$.

This procedure is shown in Figure 8.2.2. Figure 8.2.2(a) shows the samples of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. Without loss of generality, we set $\boldsymbol{v}_1 = (0, 1)^T$, since $v_{11} \in \mathbb{S}^0 = \{-1, 1\}$. Now we have to map this starting point, such that we get two vectors $\boldsymbol{v}_1'$ and $\boldsymbol{v}_2'$ satisfying $\boldsymbol{v}_1' \perp \boldsymbol{v}_2'$[4]. The Householder reflection that transforms $\boldsymbol{e}_1$ to $\boldsymbol{v}_2$ also "lifts up" $\boldsymbol{v}_1$ such that $\boldsymbol{v}_2 \perp \boldsymbol{H}_2\boldsymbol{v}_1$. Subfigure 8.2.2(b) shows the plane doing that. Subfigure 8.2.2(c) shows the projected vector $\boldsymbol{v}_1' = \boldsymbol{H}_2\boldsymbol{v}_1$ and the final Figure 8.2.2(d) emphasise the orthogonality between $\boldsymbol{v}_2' = \boldsymbol{v}_2$ and $\boldsymbol{v}_1' = \boldsymbol{H}_2\boldsymbol{v}_1$ in blue.

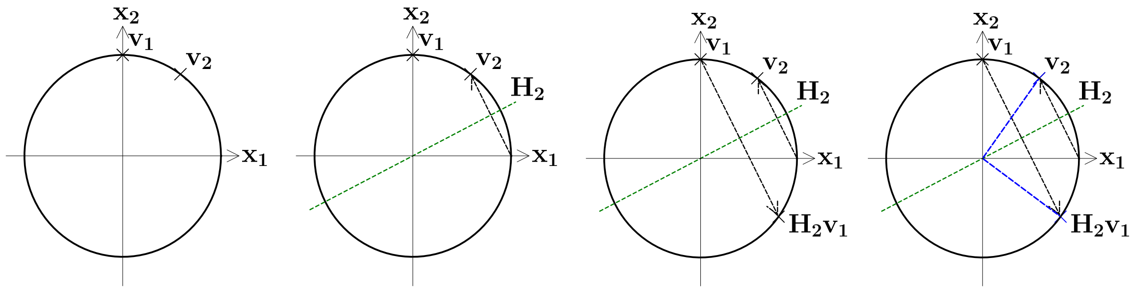
8.3 Unique PPCA

Now we have all the ingredients for building the model for PPCA that breaks the rotational symmetry, which plaques the original formulation.

8.3.1 Implementation

We start by sampling from the $\boldsymbol{v}$s from the uniform sphere. The samples from the unit sphere $\mathbb{S}^{d-1}$ are most easily obtained by drawing an i.i.d. standard Gaussian vector of dimension d and normalizing its length. This, however, increases the number of parameters by $D - Q + 1$, because we need one additional parameter for each $\{\boldsymbol{v}_d\}_{d=D-Q+1}^{D}$ compared

[4] They do not only have to be perpendicular, but also need the property that the resulting matrix $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2)^T$ is Haar distributed. As already discussed, this is fulfilled by the construction of the Householder mapping, which makes them perpendicular.



(a) one sample of $\boldsymbol{v}_1, \boldsymbol{v}_2$ (b) construction of $\boldsymbol{H}_2$ (c) mapping of $\boldsymbol{v}_1$ (d) orthogonal vectors

Figure 8.2.2: Illustration of the construction of $\boldsymbol{U} \in \mathbb{R}^{2\times 2}$.

to the $d-1$ dimensionality of $\mathbb{S}^{d-1}$ [5]. At the same time, the overparameterization helps the sampler to move around the sphere effectively. Note that the Gaussian vector $\boldsymbol{v}$ is not normalized. Normalization is not required by the Householder transformations, since Equation (8.2.14) normalizes the vector $\boldsymbol{u}$, that is used to construct the Householder transformation, anyway.

Then, the Gaussian vectors $\{\boldsymbol{v}_d \in \mathbb{R}^d\}_{d=D-Q+1}^{D}$ are transformed to Householder transformations $\{\boldsymbol{H}_d\}_{d=D-Q+1}^{D}$ via Equation (8.2.13) and Equation (8.2.15). Their product (Equation 8.2.12) results in a sample $\boldsymbol{U} \in \mathbb{R}^{D\times Q}$ from the Haar distribution. Next, we sample the ordered singular values according to the joint distribution (Equation 8.2.6) and construct $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_Q)$.

Finally, we construct $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}$ and evaluate the likelihood Equation (8.1.1). We obtain the following generative model

$$\begin{aligned}
\boldsymbol{v}_D, \ldots, \boldsymbol{v}_{D-Q+1} &\sim \mathcal{N}(0, \boldsymbol{I}) \\
\boldsymbol{\sigma} &\sim p(\boldsymbol{\sigma}) \propto \text{ eq. (8.2.6)} \\
\boldsymbol{\mu} &\sim p(\boldsymbol{\mu}), \text{ e.g. a broad Gaussian} \\
\boldsymbol{U} &= \prod_{q=1}^{Q} \boldsymbol{H}_{D-q+1}(\boldsymbol{v}_{D-q+1}) \\
\boldsymbol{\Sigma} &= \text{diag}(\boldsymbol{\sigma}) \\
\boldsymbol{W} &= \boldsymbol{U}\boldsymbol{\Sigma} \\
\sigma_{\text{noise}} &\sim p(\sigma_{\text{noise}}) \\
\boldsymbol{Y} &\sim \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{Y}_{n,:} | \boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^T + \sigma_{\text{noise}}^2 \boldsymbol{I}\right) .
\end{aligned}$$

Note that this reparameterized (by $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$) model defines the same data distribution

[5] An overparameterization is not always undesirable. As in our case, it helps the sampler by allowing for better and smoother sampling.

as the original model with the parameter $\boldsymbol{W}$, i.e.

$$p(\boldsymbol{Y}) = \int p(\boldsymbol{Y}|\boldsymbol{W})p(\boldsymbol{W})\mathrm{d}\boldsymbol{W} = \int p(\boldsymbol{Y}|\boldsymbol{U},\boldsymbol{\Sigma})p(\boldsymbol{U})p(\boldsymbol{\Sigma})\mathrm{d}\boldsymbol{U}\mathrm{d}\boldsymbol{\Sigma}\,. \tag{8.3.1}$$

Since the data distribution is only dependent on the outer product $\boldsymbol{W}\boldsymbol{W}^T$, it is not necessary that the induced distribution on $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}$ is a Gaussian. This is also not achieved by the priors that we use for $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$. However, it is necessary that the induced distribution on $\boldsymbol{W}\boldsymbol{W}^T$ is a Wishart distribution, since the data distribution is dependent only on $\boldsymbol{W}\boldsymbol{W}^T$. And this is achieved by our prior choices (Haar distribution for $\boldsymbol{U}$ and $p(\boldsymbol{\sigma})$ (Equation 8.2.6) for $\boldsymbol{\Sigma}$).

As many other model, our model also has a combinatorial symmetry introduced by the ambiguity of the eigenvector matrix $\boldsymbol{U}$. An eigenvector multiplied by -1 is still an eigenvector to the same eigenvalue. All these different modes are equivalent. So, in order to compare results across different modes, we postprocess each sample such that the first entry of each column of $\boldsymbol{U}$ is positive. In the next section, we illustrate our model on synthetic data set as well as on real world data sets.

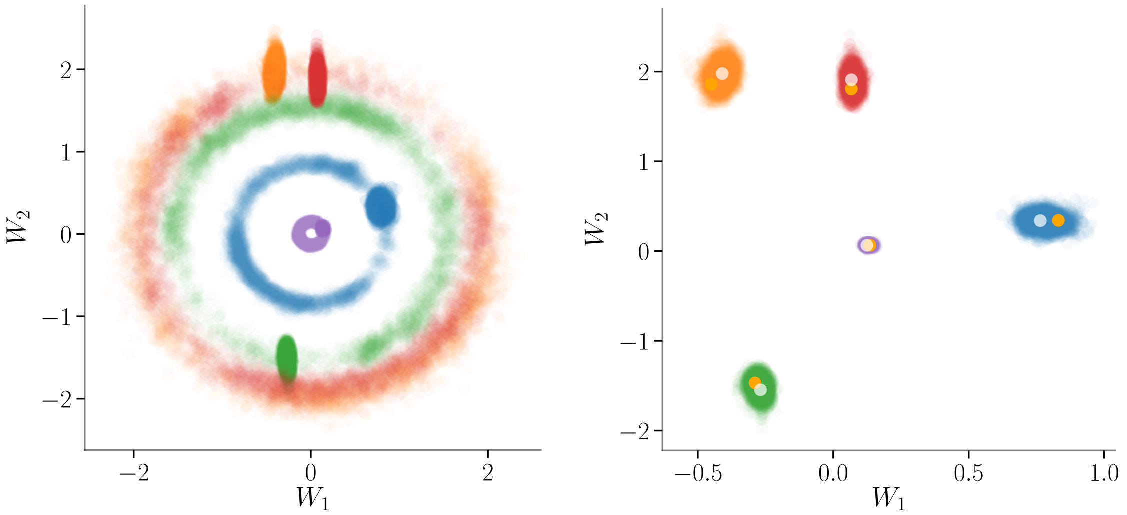
8.3.2 Model comparison

Synthetic data set

First, we build our own synthetic data set with known parameter values. The goal is to reconstruct the values of these parameters. We set $(N, D, Q) = (150, 5, 2)$ and sample $\boldsymbol{X}$ from a standard Gaussian. Then, we draw a sample $\boldsymbol{U}$ from the Stiefel manifold with Haar measure and set the singular values $(\sigma_1, \sigma_2) = (3.0, 1.0)$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2)$. The mapping $\boldsymbol{W}$ is given by $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}$. To get the noisy observations, we sample $\boldsymbol{\epsilon}$ from a zero mean Gaussian with variance 0.01 and get the data by evaluating $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}^T + \boldsymbol{\epsilon}$.

Figure 8.3.1 shows samples from the posterior distribution $p(\boldsymbol{W}|\boldsymbol{Y})$. Figure 8.3.1(a) compares the samples for the mapping $\boldsymbol{W}$ that we get by sampling $p(\boldsymbol{W}|\boldsymbol{Y})$ directly (standard approach) with our suggested approach, where we sample $p(\boldsymbol{U}, \boldsymbol{\Sigma}|\boldsymbol{Y})$ instead and calculate the mapping via $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}$. Here, we clearly see the rotational symmetry in the samples of the standard approach (rings in lighter color). As expected, our suggested approach can break the symmetry. The samples are shown in a darker color. Note that the samples of the suggested approach lie on top of the samples of the standard approach. Figure 8.3.1(b) compares the samples of the suggested approach (colored clouds) with the classical PCA solution (white dots) and the true parameters (orange dots). As expected, with the low observation noise, both, our suggested approach and the classical PCA, can reconstruct the true parameter values. However, in contrast to the classical PCA, our Bayesian approach provides a distribution for the parameters, accessing estimation uncertainty of the solution as well.

Figure 8.3.2 shows the posterior distribution of the singular values (σ_1, σ_2) along with the solution of the classical PCA (vertical solid black lines) and the true values (dashed lines). Again, our suggested approach can reconstruct the true values and the classical



(a) Sampling with the standard parameterization (rings in lighter colors) vs Householder parameterization (dense clouds in darker colors).

(b) Comparison of suggested Householder model (clouds) and the classical PCA solution (white dots) and true values (orange dots).

Figure 8.3.1: Posterior distribution $p(\boldsymbol{W}|\boldsymbol{Y})$ of the mapping $\boldsymbol{W}$ for the synthetic data set. 4000 samples for each row of $\boldsymbol{W}$ are shown in different colors (Nirwan and Bertschinger, 2019a).

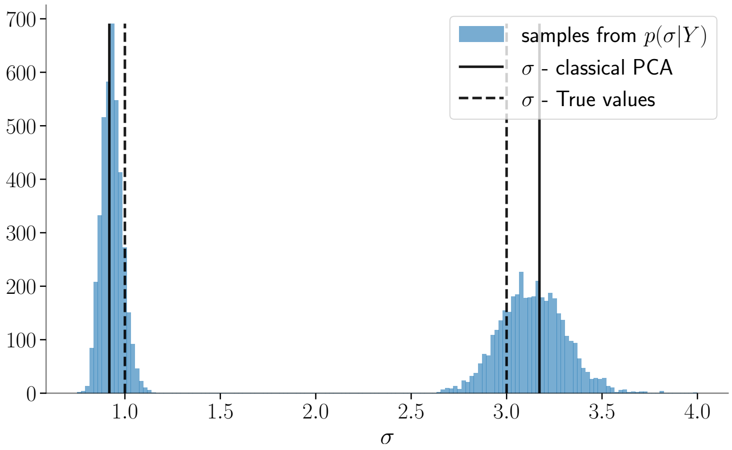


Figure 8.3.2: Posterior distribution $p(\boldsymbol{\Sigma}|\boldsymbol{Y})$ of the singular values in blue. The vertical solid black lines are the classic PCA solutions $(3.17, 0.92)$ and dashed lines are the true values $(3.0, 1.0)$.

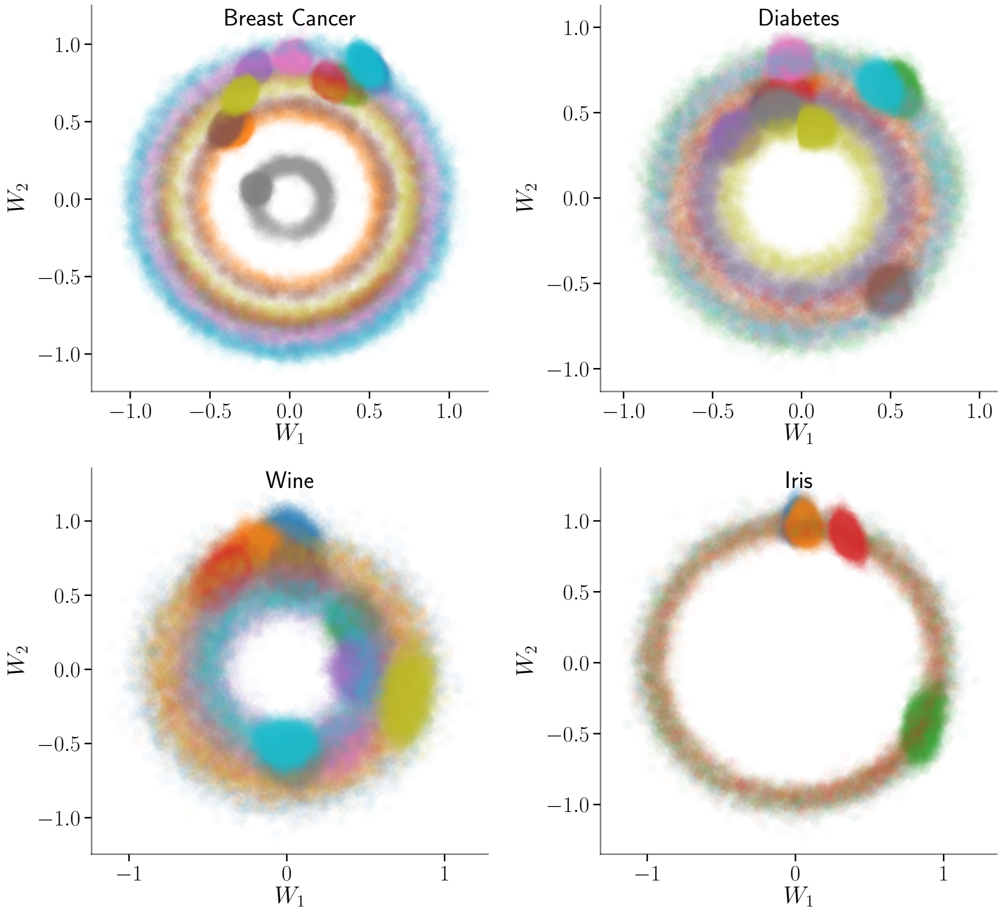


Figure 8.3.3: Sampling with standard (rings in lighter colors) and Householder parameterization (dense clouds in darker colors) for several data sets. The name of the data set is contained in the title of the subfigure.

PCA solution and in contrast to the classical PCA, it also provides the uncertainty of the parameter estimation.

Other data sets

We tested the model also on various different data sets. The name and the size of the data sets are shown in Table 8.1. All the data sets are downloaded from the Python toolbox scikit-learn (Pedregosa et al., 2011). We neglected the labels and took the feature matrix of size $N \times D$ as the input to our model. We fixed $Q = 2$ for visualisation purposes.
The results are shown in Figure 8.3.3. Each subfigure shows the results for a different data set. We plotted the posterior samples of the mapping $\boldsymbol{W}$ for the standard parameterization (directly sampling $\boldsymbol{W}$) in the background and the suggested parameterization (sampling $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ and then converting them to $\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}$) in the foreground. The result is the same

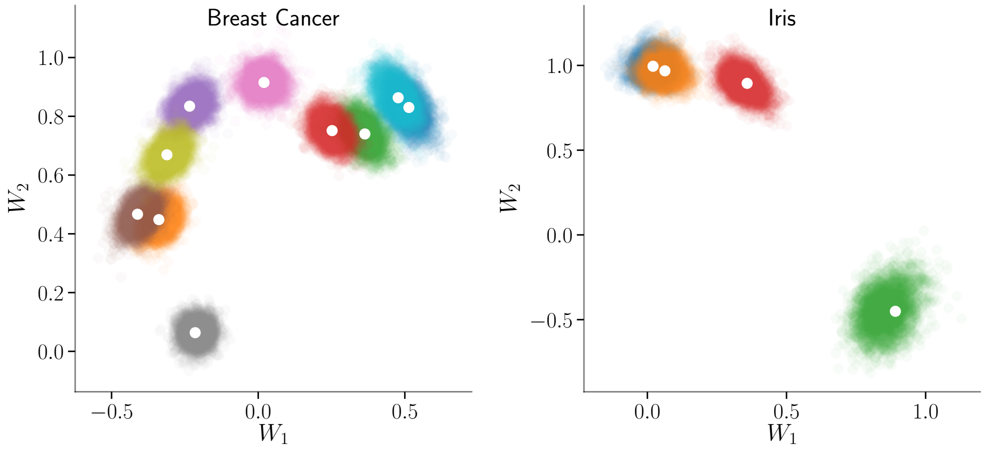


Figure 8.3.4: Comparison of Householder parameterization with classical PCA.

for all data sets: with the suggested parameterization we are able to break the rotational symmetry and uniquely identify the mapping. Figure 8.3.4 compares the solution of the suggested parameterization with the solution of the standard PCA (shown in white dots) for the breast cancer and the iris data set. As for the synthetic data set, we are also here able to reconstruct the PCA solution and provide uncertainties to the parameter estimates.

8.3.3 Computational complexity and runtime analysis

Comparing to probabilistic PCA, we compute the $D \times Q$ loading matrix $\boldsymbol{W}$ from parameters $\boldsymbol{v}$ for the principle rotation $\boldsymbol{U}$ and from $\boldsymbol{\sigma}$ for the singular value matrix $\boldsymbol{\Sigma}$. For that we use the Householder transformation. The total computational cost for that is in $\mathcal{O}(DQ^2)$ (Shepard et al., 2014). The same complexity applies to the computation of the gradients with respect to the model parameters. After obtaining $\boldsymbol{W}$, in both models (PPCA and our parameterization), the evaluation of the likelihood (Equation 8.1.1) is of order $\mathcal{O}(D^3)$. The most expensive part of the evaluation is the inversion of the covariance $\boldsymbol{W}\boldsymbol{W}^T + \sigma^2_{\text{noise}}\boldsymbol{I}$. In total, the Householder parameterization does not increase the total complexity of the model.

However, since the rotational symmetry is removed, the adaptive NUTS sampler (a version of Hamiltonian Monte Carlo, which is implemented in stan) needs fewer leap-frog steps per sample. Numerically, this leads to slightly lower wall clock times overall. Therefore, our implementation is as efficient as other sampling methods proposed for probabilistic PCA.

Table 8.1 compares the runtime for sampling 1000 samples from the posterior with Householder parameterization with the runtime of the standard parameterization on the four different data sets. Most of the time, the Householder parameterization is faster than the standard parameterization.

Table 8.1: Runtime for sampling 1000 samples from the posterior with the Householder parameterization $(\boldsymbol{U}, \boldsymbol{\Sigma})$ and the standard parameterization $\boldsymbol{W}$ for different data sets.

Model	Breast Cancer	Diabetes	Wine	Iris
Data set size (N, D)	$(569, 30)$	$(442, 10)$	$(178, 13)$	$(150, 4)$
Runtime Householder [min]	9.5	8.9	0.4	0.2
Runtime Standard [min]	25.6	2.4	1.2	0.8

8.3.4 Problems

One of the problems that shows up during sampling is caused by the closeness of the eigenvalues. If two eigenvalues σ_i, σ_{i+1} are too close to each other, such that their densities overlap, the sampler is not able to switch. This is due to the ordering constraint on the eigenvalues (see theorem 8.2.1), which also implies an ordering of the eigenvectors. If two eigenvalues σ_i and σ_{i+1} switch, their corresponding eigenvectors of dimension D have to switch simultaneously as well. This huge jump in the parameter space will mostly likely not happen, and thus ignores some of the posterior mass.

Another problem that can occurs during sampling is caused by the signs of the Householder transformations (Equation (8.2.13) and Equation (8.2.14)). The orthogonal matrix $\boldsymbol{U}$ is created by sampling $\boldsymbol{v}$s first and then transform them to the Householder transformations $\{H_d\}$. The product of the H_d's results in a sample of $\boldsymbol{U}$. This transformation involves the sign of the first element $\boldsymbol{v}_{d1}$ of the $\{\boldsymbol{v}_d \in \mathbb{S}^{d-1}\}_{d=1}^{D-Q+1}$. Therefore, a smooth change in those first elements can cause a jump in the resulting likelihood value when those elements change sign (pass 0). This is not happening for $(D, Q) = (2, 2)$, however the jump is present in the likelihood for the iris data set for $Q = 4$. In Figure 8.3.5, the samples of the mapping $\boldsymbol{W}$ are shown. Here, we see an abrupt cut in the distribution (at $\boldsymbol{W}_1 = 0$). The cut occurs because the sampler rejects all the samples on the other side because the change in the likelihood is too big compared to the change in the parameters values. As we discussed in Section 3.1.3, the acceptance probability for the next sample depends on the energy gap between the previous and the suggested sample. Thus, if the likelihood value has a huge jump when evaluated at the suggested sample, the sample is very likely to get rejected.

8.3.5 Interpretation of the parameters and other than Gaussian priors

So far, in this work, the prior has been chosen to aid comparison with the classical PCA. The SVD of $\boldsymbol{W}$ into $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ allows for a better interpretation of the parameters. $\boldsymbol{W}$ is not so easy to interpret, but $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ are. $\boldsymbol{U}$ is the rotation in the data space and $\boldsymbol{\Sigma}$ are the singular values. For the Gaussian prior on $\boldsymbol{W}$, they are also independent. Furthermore, $\boldsymbol{U}$ is uniformly distributed in the space of orthogonal matrices (Haar distribution on the Stiefel manifold).

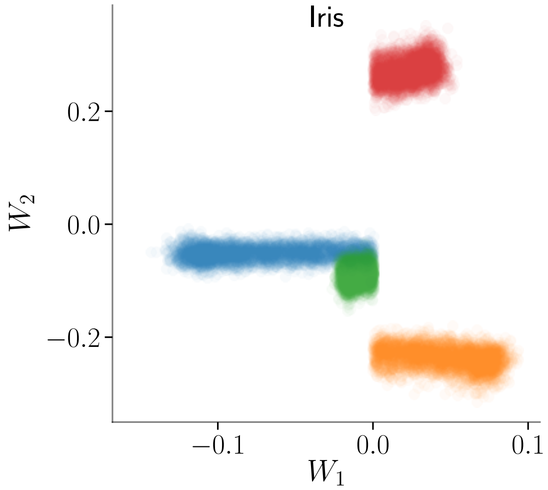


Figure 8.3.5: Illustration of the jump in the likelihood and their effect on the samples of the parameters $\boldsymbol{W}$. The sampler cannot pass $\boldsymbol{W}_1 = 0$ because of that jump.

Other priors of this structure can be constructed very easily. If one knows a-priori the structure of the data, the information can be embedded into the priors. E.g., information about principle variances can be encoded into the prior of $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_Q)$. Also, information about preferred directions/rotations of the data can be encoded into the prior of the rotation matrix $\boldsymbol{U}$. E.g. automatic relevance determination is easily accomplished by putting a shrinkage/sparsity prior on the principle components.

8.4 Extension to nonlinear models

We discussed the nonlinear extension to dual probabilistic PCA (PPCA), the Gaussian process latent variable model (GPLVM), in Section 5.3. In dual PPCA, instead of the marginalization of the latent space $\boldsymbol{X}$, we marginalize the mapping $\boldsymbol{W}$ and end up with the following marginal likelihood

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{d=1}^{D} \mathcal{N}(\boldsymbol{Y}_{:,d}|\boldsymbol{0}, \boldsymbol{K} + \sigma^2 \boldsymbol{I}) \;, \tag{8.4.1}$$

where $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{K}_{ij} = \boldsymbol{X}_{i,:}^T \boldsymbol{X}_{j,:}$. GPLVM generalizes that model (Equation 8.4.1) by replacing the linear kernel $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$ with a nonlinear one, e.g. the RBF

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_{\mathrm{RBF}}^2 \exp\left\{ -\frac{1}{2l^2} \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 \right\} \;, \tag{8.4.2}$$

where σ_{RBF}^2 is the kernel variance and l is the kernel length scale. The likelihood with the new kernel has still the rotational symmetry. This is the case with all stationary kernels.

They are only distance-dependent and since rotations preserve the distance between points, they all have the rotational symmetry. The question is now: can we, by reparameterizing the model again with the Householder parameterization, break the symmetry in nonlinear models as well?

To test our model in the nonlinear case, we reparameterize the latent space $\boldsymbol{X}$ by $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ as we did in the PPCA case with the mapping $\boldsymbol{W}$. Then we sample $\boldsymbol{v}$ and $\boldsymbol{\sigma}$, transform them to $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ and reconstruct $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}$ as described earlier. Figure 8.4.1 shows the posterior samples for the GPLVM with RBF kernel (Equation 8.4.2) applied to the Breast Cancer Wisonsin data set. The top row shows the samples for three different chains from the standard parameterization, where we sample $\boldsymbol{X}$ directly. Here, we see the rotational symmetry in the posterior implied by the likelihood and the Gaussian prior on $\boldsymbol{X}$. The bottom row shows the samples for three different chains from the Householder parameterization, where we sample $\boldsymbol{v}$ and $\boldsymbol{\sigma}$, transform them to $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ and reconstruct $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}$. In the bottom row we see no rotational symmetry in the latent space $\boldsymbol{X}$ anymore. However, due to the increased complexity and flexibility induced by the non-linear kernel, many local minima arise and the chains with both (the standard and the Householder) parameterizations converge to different minima.

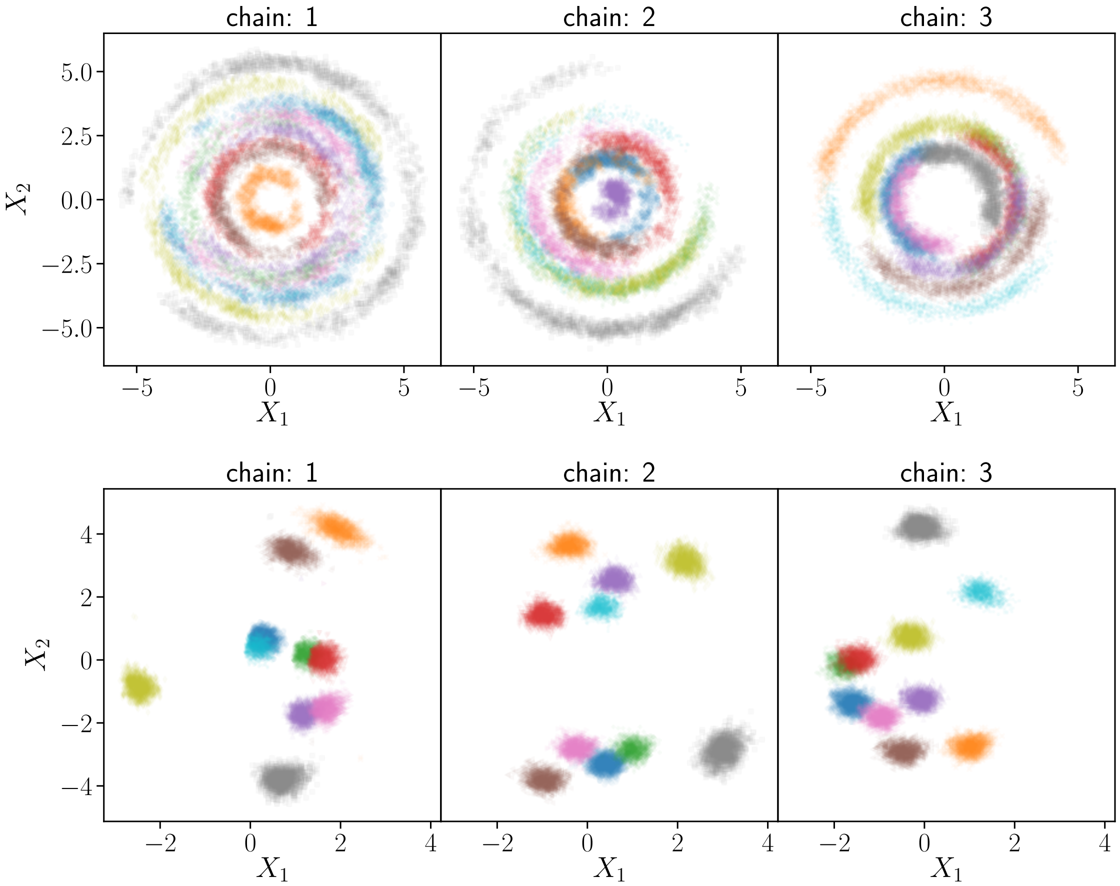


Figure 8.4.1: Posterior samples of the latent space $\boldsymbol{X}$ for the GPLVM with standard (top row) and Householder parameterization (bottom row) from different chains.

Chapter 9

Order Statistics

The focus of this and the next chapter is to train a model using aggregated data. Many times there is no information about individual samples form a data set, but just some quantiles, like the median, 25% or 75% quantiles are available. Especially, with more and more regulations with the data protection laws, getting sensitive individual data is hard, e.g.: in the medical domain or in the economy. Many agencies collect data which is only accessible in aggregated form. E.g.: Instead of providing individual salary data, providers only provide some statistics of the data.

In Nirwan and Bertschinger (2020), we propose a Bayesian method to learn from those statistics. Our method has a very intuitive noise model based on the order statistics of the samples, which is able to capture uncertainties better than current methods do.

In this chapter, we provide the mathematical background needed to understand our method that is based on the order statistics. We start with the order statistics of a uniform distribution and then generalize to non-uniform distributions. Next chapter provides experiments, where we use order statistics to fit synthetic data first and then apply the method on real world data sets.

9.1 Order Statistics of a Uniform distribution

In our discussion about order statistics we are going to restrict ourself to the continuous case. To fix the notation, assume that we are given n real-valued i.i.d. observations $(X_1, \dots, X_n)$ from a continuous distribution with PDF f and CDF F. We denote by $(X_{(1)}, \dots, X_{(n)})$ the ordered series of the observations, where $X_{(1)} \leq \cdots \leq X_{(n)}$. The k-th order statistic of that sample is equal to its k-th smallest value $X_{(k)}$[1]. $X_{(k)}$ itself is a random variable. The smallest value $X_{(1)}$, for example, will be different for another n samples from the same distribution with PDF f. The same is true for all the other order statistics. The k-th smallest value (k-th order statistic), therefore, has its own distribution $p(X_{(k)})$ that we are interested to model.

[1] Some examples are: $X_{(1)} = \min(X_1, \dots, X_n)$ is the minimum, $X_{(n)} = \max(X_1, \dots, X_n)$ is the maximum and for odd n $X_{((n+1)/2)} = \text{median}(X_1, \dots, X_n)$ is the median of the sample. For even n, the median, for example, can be the average of the $n/2$ and the $n/2 + 1$ order statistics.

There are many other statistics that are easily defined in terms of the order statistics. For example, the sample range $R = X_{(n)} - X_{(1)}$ is the distance between the smallest and the largest observation and is a measure of the dispersion in the sample. Another measure of the dispersion is the interquartile range, which is the distance between the lower quartile (25% quantile) and the upper quartile (75% quantile). Quantiles themselves can be expressed as order statistics. For a sample size n, the 25% quantile will be the $0.25n$-th order statistic[2].

To get an intuition of the order statistics, let's start with the uniform distribution on the interval $[0, 1]$. If $(X_{(1)}, \dots, X_{(n)})$ are samples from the Uniform$(0, 1)$, the CDF of the k-th order statistic has the following interpretation: For an event $X_{(k)} < x$, there will be at least k many $X_{(i)}$'s that are smaller or equal to x. Therefore, the drawing of all the n samples $(X_1, \dots, X_n)$ can be seen as Bernoulli trials, where the success is defined as $X_i < x \ \forall \ i \in 1, \dots, n$. The CDF for the k-th order statistic $P(X_{(k)} < x)$ is then defined as at least k successes and has the following form

$$P(X_{(k)} < x) = \sum_{i=k}^{n} \binom{n}{i} x^i (1-x)^{n-i} \ . \tag{9.1.1}$$

The PDF is given by the derivative of the CDF and takes the form

$$p(X_{(k)} = x) = n\binom{n-1}{k-1} x^{k-1} (1-x)^{n-k} \ , \tag{9.1.2}$$

which is the Beta distribution Beta$(k, n-k+1)$ and can be interpreted in the following way: Out of n possible values, one of them becomes the k-th order statistic $X_{(k)} = x$. For the $n-1$ values left, there must be exactly $k-1$ values smaller than x. The probability for one of the observations to be smaller than x for the standard Uniform distribution is just x.

So, for samples drawn from a standard Uniform distribution, the k-th order statistic $X_{(k)}$ has a Beta-distribution

$$p(X_{(k)} = x) = \text{Beta}(x|k, n-k+1) \ . \tag{9.1.3}$$

9.2 Generalization to non-Uniform distributions

In a more general case the samples are draw from a non-uniform distribution with PDF f_x and CDF F_x, instead of a Uniform distribution. To get the density of the k-th order statistic in this case, we just transform the samples such that they become uniformly distributed. This is done by applying the CDF F_x on the samples. Thus, $\{U_i = F_x(X_i)\}_{i=1}^{n}$ will correspond to samples from the standard Uniform distribution again.

For uniformly distributed observations, we already know that the k-th order statistic has the density function given by Equation (9.1.2). However, since we are transforming a

[2] The order statistics are refereed to with integer values. Therefore, if $0.25n$ is not an integer, one can take $\lfloor 0.25n \rfloor$ or $\lceil 0.25n \rceil$ as an approximation to the 25% quantile.

random variable, we have to correct for the resulting change of measure by multiplying with the absolute Jacobian[3]. So, the PDF for the k-th order statistic of observations from a PDF f_x and CDF F_x becomes

$$p_x(X_{(k)}) = p_u\left(F_x(X_{(k)})\right) \left|\frac{\mathrm{d}F_x(x)}{\mathrm{d}x}\right|_{x=X_{(k)}} , \tag{9.2.1}$$

where p_u is the PDF of the k-th order statistic according to the uniform distribution (Beta-distribution)

$$U_{(k)} = F_x(X_{(k)}) \sim \mathrm{Beta}\left(F_x(X_{(k)})|k, n-k+1\right) := p_u(U_{(k)}) \tag{9.2.2}$$

and the last term is the Jacobian correction. Using Equation (9.2.2) and $|\mathrm{d}F_x(x)/\mathrm{d}x| = f_x(x)$, we obtain

$$p_x(X_{(k)}) = \mathrm{Beta}\left(F_x(X_{(k)})|k, n-k+1\right) f_x(X_{(k)}) , \tag{9.2.3}$$

which is the k-th order statistic of any random variable X with PDF f_x and CDF F_x.

9.3 Joint distribution of Order Statistics

For a sample size of n values, we denote as $\boldsymbol{x} \in \mathbb{R}^M$ the observed/sample quantile values for M quantiles. Even if the observed n samples are i.i.d., their order statistics is not independent

$$p_x(X_{(1)}, \ldots, X_{(n)}|\theta) \neq \prod_{k=1}^{n} p_x(X_{(k)}|\theta) . \tag{9.3.1}$$

The same is true for their M sample quantiles

$$p_x(X_{(1)}, \ldots, X_{(M)}|\theta) \neq \prod_{k=1}^{M} p_x(X_{(k)}|\theta) . \tag{9.3.2}$$

This is due to their ordering constraint $X_{(1)} \leq \cdots \leq X_{(n)}$. Knowing the value of $X_{(k)}$, for example, forces a constraint on all the other ones. The values of $\{X_{(i)}\}_{i=1}^{k-1}$ have to be smaller and $\{X_{(i)}\}_{i=k+1}^{n}$ have to be greater than the observed value of $X_{(k)}$.

To derive the joint distribution of the full order statistics, we start with the joint distribution of two such quantile observations. The application of the CDF F to all of the observed quantile values $\boldsymbol{x}$, leads to a uniformly distributed random vector $\boldsymbol{U} = (U_1, \ldots, U_M)$. The joint PDF of two order statistics $U_{(i)}$ and $U_{(j)}$, where $U_{(i)} < U_{(j)}$ then takes the following form

$$p_{U_{(i)},U_{(j)}}(u, v) = n! \frac{u^{i-1}}{(i-1)!} \frac{(v-u)^{j-i-1}}{(j-i-1)!} \frac{(1-v)^{n-j}}{(n-j)!} , \tag{9.3.3}$$

[3] Given a probability $p_u(u)$ on u and an invertible map g, such that $u = g(x)$, the density $p_x(x)$ on x is given by $p_x(x) = p_u\left(g(x)\right) \left|\frac{dg(x)}{dx}\right|$.

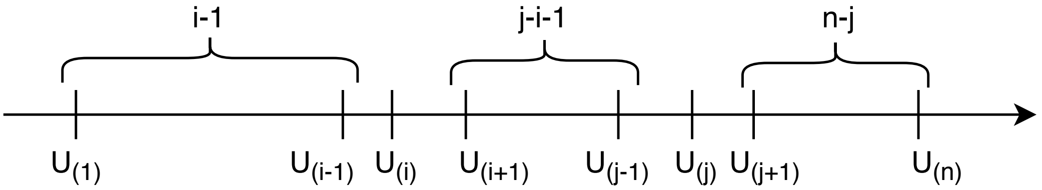


Figure 9.3.1: Illustration of the PDF of the joint order statistics (Equation 9.3.3).

where u and v correspond to the observed values of $U_{(i)}$ and $U_{(j)}$. Figure 9.3.1 illustrates Equation (9.3.3). The first fraction $u^{i-1}/(i-1)!$ is proportional to the binomial distribution of $i-1$ of the samples being smaller than u. The second term $(v-u)^{j-i-1}/(j-i-1)!$ corresponds to $j-i-1$ of the samples being in the interval (u,v) and the last term $(1-v)^{n-j}/(n-j)!$ corresponds to $(n-j)$ samples being greater than v.
For $M>2$, we have to split the interval $[0,1]$ into $M+1$ pieces and apply the same logic as depicted in Figure 9.3.1. Therefore, for M observed quantile values $\boldsymbol{u}=(u_1,\ldots,u_M)$ (with total sample size n) from a uniform distribution with their order $\boldsymbol{k}=(k_1,\ldots,k_M)\in\mathbb{N}^M$, their joint PDF becomes

$$p_{\boldsymbol{U}}(\boldsymbol{u}|\boldsymbol{k}) = \; c u_1^{k_1-1}(1-u_M)^{n-k_M}\prod_{m=2}^{M}(u_m-u_{m-1})^{k_m-k_{m-1}-1}\,, \tag{9.3.4}$$

where the normalization constant c is given by (See Appendix A.1)

$$c=\frac{n!}{(k_1-1)!(n-k_m)!\prod_{m=2}^{M}(k_m-k_{m-1}-1)!}\,. \tag{9.3.5}$$

The extension to the joint PDF of the order statistics for samples from a non-uniform distribution with PDF f and CDF F is again straight forward. As already mentioned in the last section, we apply F to the samples, which will convert them to the desired samples from a uniform distribution. For the samples from the uniform distribution, we can use the PDF for the order statistics for a uniform distribution (Equation 9.3.4) adjusted with the Jacobian correction.
So, for observations $\boldsymbol{x}\in\mathbb{R}^M$ from a PDF f, their corresponding order $\boldsymbol{k}\in\mathbb{N}^M$ and the total number of observations n, we get the following joint order statistics

$$\begin{aligned} p_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{k}) = {} & cF(x_1)^{k_1-1}(1-F(x_M))^{n-k_M} \\ & \prod_{m=2}^{M}(F(x_m)-F(x_{m-1}))^{k_m-k_{m-1}-1}\prod_{m=1}^{M}f(x_m)\,. \end{aligned} \tag{9.3.6}$$

Chapter 10

Bayesian quantile matching estimation

As stated in the last chapter, we want to use order statistics of a sample to match observed quantiles to a distribution.
There are other methods for matching the quantiles to a distribution. But all of them have some disadvantages. The Federal Institute for Risk Assessment in Germany has open sourced an R-package (Belgorodski et al., 2017) that fits the quantiles by minimizing the mean squared error (MSE) between the cumulative density function (CDF) and the empirical quantile values. As we discuss in this chapter, this has many downsides and is not recommended to do. Similarly, Sgouropoulos et al. (2015) and Dominicy and Veredas (2013) propose to minimize the quadratic distance between some quantile statistics of modeled and actual data.
In this chapter we start by illustrating the problem and how to solve it using order statistics that we introduced in Chapter 9. We define the problem in Section 10.1. In Section 10.2, we present the solution to the problem using order statistics. Section 10.3 briefly introduces another model (CDF regression with Gaussian noise) that is frequently used in these problem settings. In the experiments in Section 10.4, we conduct experiments on synthetic data and compare the CDF regression with Gaussian noise (equivalent to MSE minimization) with our suggested method based on order statistics. At the end of this chapter we fit our model to real world salary data of several countries and suggest a procedure for model selection. The work in this chapter can be found on arxiv (Nirwan and Bertschinger, 2020).

10.1 Problem definition

Suppose there are N samples drawn from a distribution p but we have no access to those samples. We are only provided some aggregated information about the samples. The information we have might be M quantiles $\boldsymbol{q} = (q_1, \dots, q_M)$ and their corresponding empirical values $\boldsymbol{x} = (x_1, \dots, x_M)$. The task is to infer the underlying distribution from this information. Note that the quantile values $\boldsymbol{x}$ are random variables for fixed quantiles

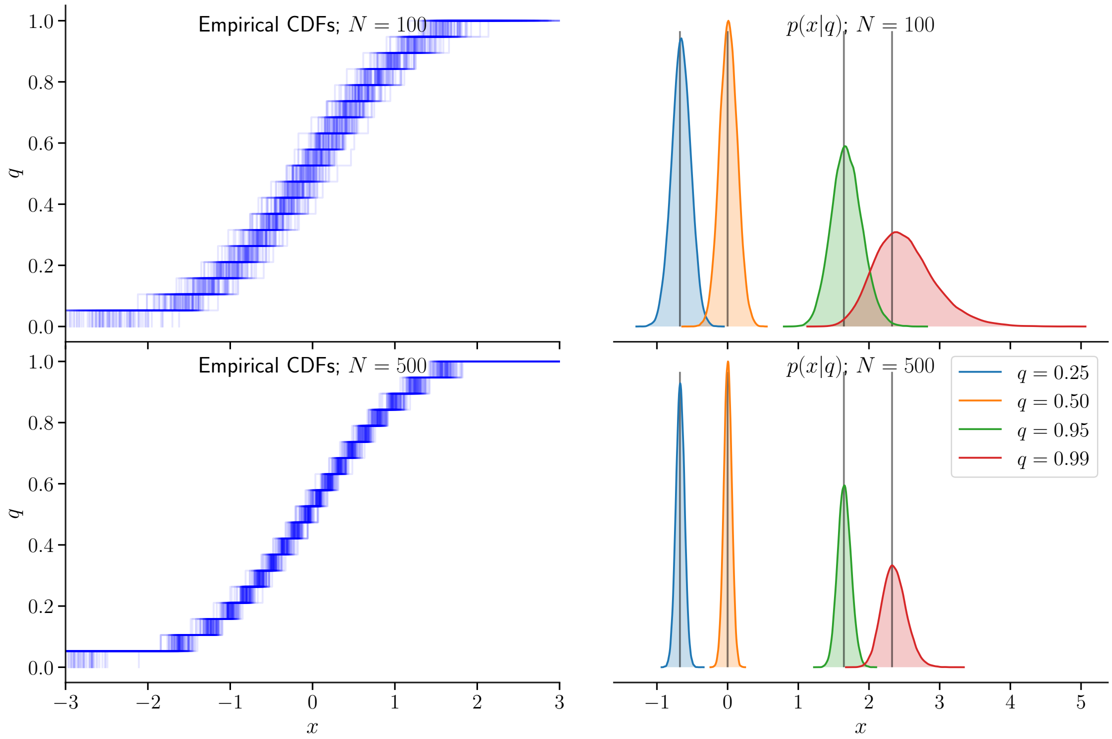


Figure 10.1.1: Illustration of the effect of the underlying sample size N and the quantile q on the uncertainty of the quantile value x.

$\boldsymbol{q}$. Another N samples from the same distribution p will result in different quantile values $\boldsymbol{x}$. That means, the quantile values $\boldsymbol{x}$ themselves are noisy and need to be treated as such.

Intuitively, for a large sample size N, the variance in the quantile values $\boldsymbol{x}$ will not be as big as for a small sample size. Also, the variance in the values of the tail quantiles is, intuitively, larger than the variance in the values of quantiles in the center, because fewer samples are available at the tails than at the center of the distribution for a fixed N. This intuition is shown empirically in Figure 10.1.1, where the top row shows plots for $N = 100$ and the bottom row for $N = 500$. The left column shows 100 CDFs based on 20 empirical quantile values for equidistant quantiles in interval $[0, 1]$ calculated from N draws from a standard Gaussian distribution and the right column shows the distribution $p(x|q)$ of the quantile value x for $q = 0.25, 0.50, 0.95$ and 0.99. The distributions on the right correspond to the cut of the CDFs on the left for the corresponding quantile q. Here, the uncertainty of the quantile value x due to the underlying sample size N and the specific quantile q becomes obvious and is reflected in the width of the distribution $p(x|q)$. This difference in the uncertainty of different quantiles q and sample size N is exactly what we want our model to capture. The marginal $p(x|q)$ and the joint distribution $p(\boldsymbol{x}|\boldsymbol{q})$ for observations $\boldsymbol{x}$ have been derived in Chapter 9 and are given by Equation (9.2.3) and Equation (9.3.6) respectively. In the next section, we use these equations to fit a non-uniform distribution given only the quantile information.

10.2 Fit a non-Uniform distribution given quantile information

In Section 9.3, we derived the joint distribution of the order statistics of observations $\boldsymbol{x} \in \mathbb{R}^M$ using their corresponding order $\boldsymbol{k} \in \mathbb{N}^M$. In most settings not the order $\boldsymbol{k}$ of M observations based on an actual sample size of N data points is given, but information about the corresponding quantiles $\boldsymbol{q}$. Given M observed quantiles $\boldsymbol{q} = (q_1, \ldots, q_M) \in [0,1]^M$ and the number of the samples size N the observation is based on, we can calculate the order $\boldsymbol{k}$. $\boldsymbol{k}$ is simply the product of a quantile q_i and the number of total sample size N, i.e.: $k_i = q_i N$ [1].

10.2.1 Joint distribution of observed quantile values

Thus, we can rewrite the joint distribution (Equation 9.3.6) in terms of the observed quantiles $\boldsymbol{q}$ instead of the ordering $\boldsymbol{k}$. For an underlying sample size N, M observed quantiles $\boldsymbol{q} = (q_1, \ldots, q_M) \in [0,1]^M$ and their corresponding values $\boldsymbol{x} = (x_1, \ldots, x_M) \in \mathbb{R}^M$, the model likelihood becomes

$$\begin{aligned} p_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{q},\boldsymbol{\theta},N) = {} & cF_{\boldsymbol{\theta}}(x_1)^{q_1 N-1}(1-F_{\boldsymbol{\theta}}(x_M))^{N-q_M N} \\ & \prod_{m=2}^{M}(F_{\boldsymbol{\theta}}(x_m)-F_{\boldsymbol{\theta}}(x_{m-1}))^{q_m N-q_{m-1}N-1} \\ & \prod_{m=1}^{M} f_{\boldsymbol{\theta}}(x_m)\ , \end{aligned} \tag{10.2.1}$$

where $F_{\boldsymbol{\theta}}$ is the CDF and $f_{\boldsymbol{\theta}}$ the PDF of the distribution we want to fit to $(N, \boldsymbol{q}, \boldsymbol{x})$ parameterized by $\boldsymbol{\theta}$.

The next step is to learn a distribution that best fits the data, given the likelihood (Equation 10.2.1). This can be done by maximizing the likelihood with respect to the parameter $\boldsymbol{\theta}$ for an observation $(N, \boldsymbol{q}, \boldsymbol{x})$ or assume a prior $p(\boldsymbol{\theta})$ for the parameters and infer the posterior $p(\boldsymbol{\theta}|N, \boldsymbol{x}, \boldsymbol{q})$ given by Bayes rule

$$p(\boldsymbol{\theta}|N,\boldsymbol{x},\boldsymbol{q}) = \frac{p(\boldsymbol{x}|\boldsymbol{q},\boldsymbol{\theta},N)p(\boldsymbol{\theta})}{p(\boldsymbol{x})}\ , \tag{10.2.2}$$

where $p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{q},\boldsymbol{\theta},N)p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$ is the marginal likelihood. Since the posterior is analytically intractable, we need to resort to approximation methods e.g. MCMC or variational Bayes. For the experiments in this chapter, we used MCMC (NUTS version of HMC, as explained in Section 3.1.3) implemented in the probabilistic programming language Stan (Carpenter et al., 2017).

[1] Note that k_i is a positive integer but q_i can be any number between 0 and 1. Thus, by not setting k_i as $\lfloor q_i N \rfloor$ or $\lceil q_i N \rceil$ but as $k_i = q_i N$ we are interpolating.

10.2.2 Generative model

Given $(N, \boldsymbol{q}, \boldsymbol{x})$ and the PDF $f_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\theta}$, we obtain the following generative model

$$\begin{aligned} &\boldsymbol{\theta} \sim p(\boldsymbol{\theta}) \\ &\boldsymbol{x} \sim p_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{q}, \boldsymbol{\theta}, N) \ , \end{aligned} \tag{10.2.3}$$

where $p(\boldsymbol{\theta})$ is the prior distribution of the parameters of the PDF $f_{\boldsymbol{\theta}}$ and the likelihood $p_{\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{q}, \boldsymbol{\theta}, N)$ is given by Equation (10.2.1). For the most practical purposes and for all examples presented in this chapter, i.e. where M is small, the code runs almost instantaneously. In particular, by Equation (10.2.1), computing the likelihood has linear complexity requiring $\mathcal{O}(M)$ CDF and PDF evaluations.

10.2.3 Model selection

One part of the model is the choice of the PDF f that we want to use to describe the data. There are many choices for f and each leads to a different model and different fit. Therefore, it would also be useful to know, how to select f. One measure for the goodness of how well the data is described by the model is the likelihood score. We take the distribution f that maximizes the likelihood (Equation 10.2.1).

We fit the data by using Hamiltonian Monte Carlo in Stan, which builds a chain of parameter values that results to samples from the posterior. While building this chain, we also calculate the likelihood for each of the parameter set. Thus, we get the likelihood values of the typical set. These values can be compared for different fs and we can take the f that has the best likelihood values. An easy method would be to compare the mean of the distributions and take the model that maximizes the mean. For the model selection in Section 10.5, we will compare the mean of the likelihood values for different distributions and take the one that maximizes the mean.

10.3 CDF regression model

The joint distribution $p(\boldsymbol{x}|\boldsymbol{q})$ (Equation 10.2.1) is an assumption for the noise model that arises naturally from the ordering of the observed values. But, there are also other models for the noise that are frequently used. One of them is the Gaussian noise model. As stated in the introduction, this model corresponds to the MSE minimization. Given the quantiles $\boldsymbol{q}$ and the value at the quantiles $\boldsymbol{x}$, the idea is to choose a parametric form of the distribution $f_{\boldsymbol{\theta}}$, parameterized by $\boldsymbol{\theta}$, and find the parameters such the the MSE is minimized

$$\min_{\boldsymbol{\theta}} \sum_{m=1}^{M} (q_m - F_{\boldsymbol{\theta}}(x_m))^2 \ . \tag{10.3.1}$$

The probabilistic analogue to the MSE is the Gaussian noise model, where the likelihood is a Gaussian and factorizes conditioned on the parameters $\boldsymbol{\theta}$. Thus, we can equivalently consider the maximum likelihood estimate for the parameters $\boldsymbol{\theta}$ for a model with the following likelihood

$$p(\boldsymbol{q}|\boldsymbol{\theta}, \boldsymbol{x}, \sigma^2_{\text{noise}}) = \prod_{m=1}^{M} \mathcal{N}\left(\boldsymbol{q}_m | F_{\boldsymbol{\theta}}(\boldsymbol{x}_m), \sigma^2_{\text{noise}}\right) \ . \tag{10.3.2}$$

By comparing Equation (10.3.2) and Equation (10.2.1), we see that the modeling of the uncertainty is quite different. The Gaussian noise model corresponds to a distribution $p(q|x)$ (in contrast to the order statistics $p(x|q)$), i.e. considering the quantile itself as a variable. However, in most applications, the quantiles $\boldsymbol{q}$ are chosen a-priori and their corresponding values $\boldsymbol{x}$ are estimated or reported. The Gaussian noise model also leads to a different penalty for the deviation of the regression function from the observed data points. In particular, minimizing the MSE or maximizing the Gaussian likelihood penalizes the deviation just based on the distance of the observed point to the CDF at that point, without any dependency on q. This, however is obviously not true, as we saw in Figure 10.1.1 and leads to a model that underemphasizes the uncertainty of the tails massively and implicitly downweights the information coming from the samples from the tails.

10.4 Experiments

In this section we conduct experiments on quantile matching estimations on synthetic data sets and discuss the results. We also compare the fit by the order statistics (Equation 10.2.1) to the fit by the Gaussian noise model (Equation 10.3.2). We start by looking at the posterior distribution of the parameters $\boldsymbol{\theta}$ for both noise models (order statistics and the Gaussian noise model). Then we analyse the dependency of the posterior on the number of total samples N. After that we make a stability analysis of both models and at the end we look at the ability of both models to deal with unspecified models. For all the experiments in this section, we took a very broad Gaussian prior $p(\boldsymbol{\theta})$ for all parameters $\boldsymbol{\theta}$.

10.4.1 Bayesian quantile matching estimation

In our first experiment, we fit a Gaussian distribution to some synthetic data. The data are generated by taking N samples from a Gaussian with known parameters μ and σ. We draw N samples from $\mathcal{N}(\mu, \sigma)$ and take M quantile values $\boldsymbol{x} \in \mathbb{R}^M$ for given quantiles $\boldsymbol{q} \in \mathbb{R}^M$. The tuple $\mathcal{D} = (\boldsymbol{x}, \boldsymbol{q}, N)$ is the synthetic data that we learn from. The goal is to infer the right parameters μ and σ from $(\boldsymbol{x}, \boldsymbol{q}, N)$. We model that data by the order statistics and the Gaussian noise model.

Figure 10.4.1 shows the sampled posterior $p(\mu|\mathcal{D})$ and $p(\sigma|\mathcal{D})$ for both models (order statistics in blue and Gaussian noise mode in orange). The Kernel Density Estimate (KDE) of the samples is plotted and the true values of the data generating distribution are shown in with the vertical black line. We set $\mu = 2.0$ and $\sigma = 1.0$. The top row shows the results for $N = 100$ and $M = 10$ and the bottom row shows the results for $N = 500$ and $M = 100$. The $\boldsymbol{q}$ is chosen equidistantly between 5% and 95% for both plots. The

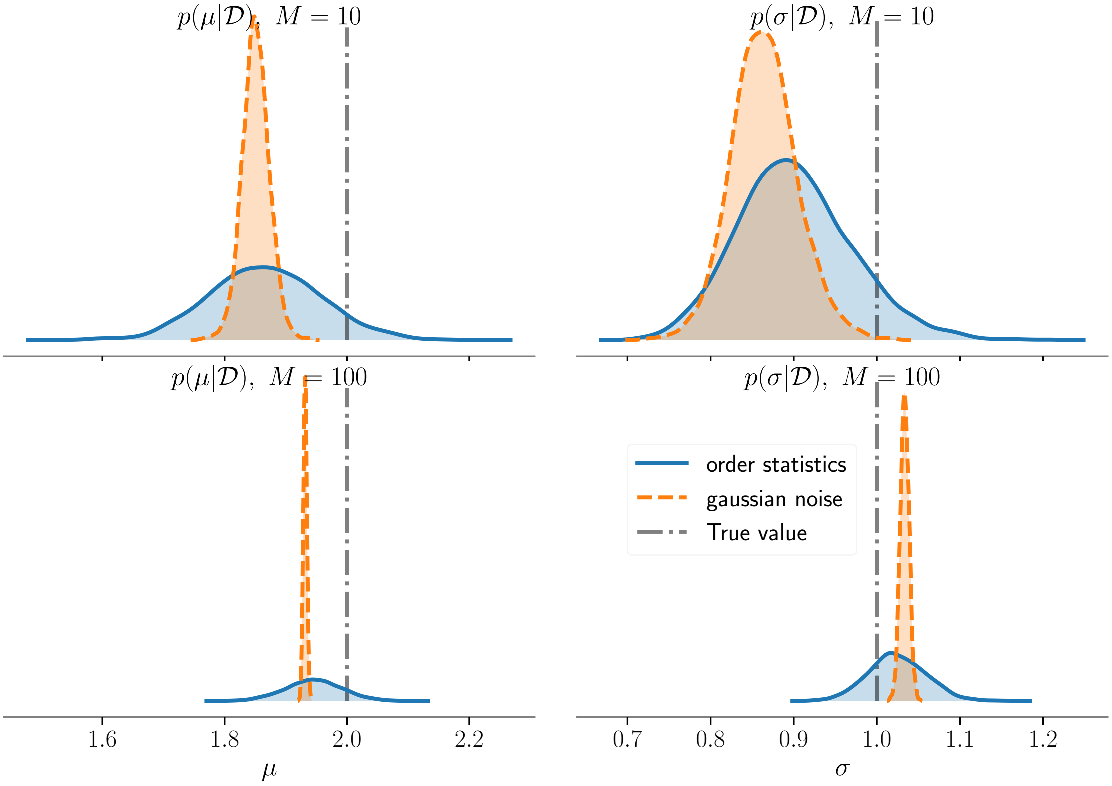


Figure 10.4.1: Posterior distribution of μ (left column) and σ (right column) for different values of M (rows) learned by the order statistics in blue and the Gaussian noise model in orange. True parameters are shown with vertical black line.

Gaussian noise model (orange) totally underestimates the uncertainty for μ compared to the order statistics. As one can see, it puts zero probability mass on the true parameters. Even if we had an outlier in the observations $\mathcal{D}$, the model should not totally exclude the true values. The fit by the order statistics, however, has a higher uncertainty and puts more mass at the true parameters. Also for higher M, the uncertainty in the posterior for both models decreases as one would expect. Note that higher uncertainty of the parameter estimates is not a bad thing. If the data cannot pinpoint the value of a parameter, we ideally want that to be reflected in its posterior distribution.

10.4.2 Dependency on N

The intuition that with a larger sample size N, the estimation of the quantile values $\boldsymbol{x}$ at $\boldsymbol{q}$ should be more accurate, is build in the model likelihood of the order statistics (Equation 10.2.1). In this subsection we will have a look into the posterior as a function of the total sample size N. Larger values of N result in more accurate estimates for $\boldsymbol{x}$. This should also be reflected in the posterior uncertainty, which should decrease (become more

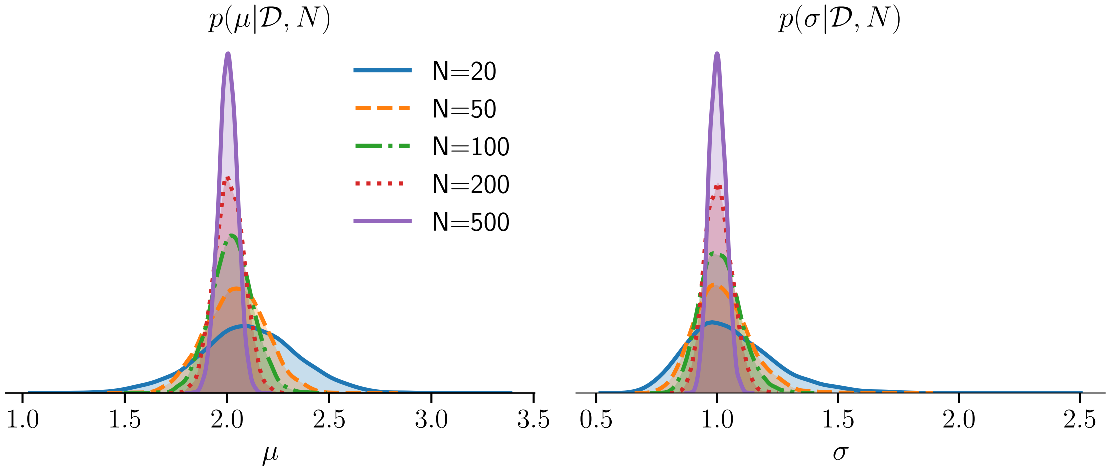


Figure 10.4.2: Posterior distribution of μ and σ for different N.

accurate) for higher values of N.

Figure 10.4.2 shows the posterior for μ and σ for several different values of N. The data $\mathcal{D} = (\boldsymbol{x}, \boldsymbol{q}, N)$ is coming again from a Gaussian $\mathcal{N}(\mu, \sigma)$ for $(\mu, \sigma) = (2.0, 1.0)$, M is set to 15 and the quantiles $\boldsymbol{q}$ are chosen equidistantly between 5% and 95%. As we expected, the higher the N the more compact the posterior becomes and puts more and more probability mass around the true values ($\mu = 2.0$ and $\sigma = 1.0$).

10.4.3 Robustness to change of a single point in the sample

In this subsection we analyse the robustness of both models to the change of a single point. We change the value of a single sample of the underlying samples and analyse the change in the fitted posterior distribution. The criteria for a robust fit is that it should not change much if we vary only one of N samples (here, we visually compare the change of both models - order statistics and Gaussian noise model), especially if the number of the total sample size N is high.

Top row of Figure 10.4.3 shows the mean of the posterior distribution for both models (dark lines) and the 5% and 95% quantiles in (light colors). The x-axis represents the value of the first sample from the total 200 samples, which were taken from a Gaussian $\mathcal{N}(\mu, \sigma)$ with $(\mu, \sigma) = (2.0, 1.0)$. M was set to 15 and $\boldsymbol{q}$ was taken equidistantly from 1% to 99%. We changed the value of the first element of the sample from $\mu - 7\sigma$ to $\mu + 7\sigma$ for both models. So, we see the change in the full posterior mass when moving a single sample from -5 to 9. The estimates by the order statistics have higher estimation uncertainty (broader posterior) than the estimates by the Gaussian noise model. The relative change in the posterior distribution of the mean $p(\mu|\mathcal{D})$ is significantly smaller for the order statistics compared to the Gaussian noise model. The absolute change, however, is slightly larger for the order statistics. The full distribution on either side is shown again in Figure 10.4.4. The solid line represents the posterior for the fit, where the first element of the sample is

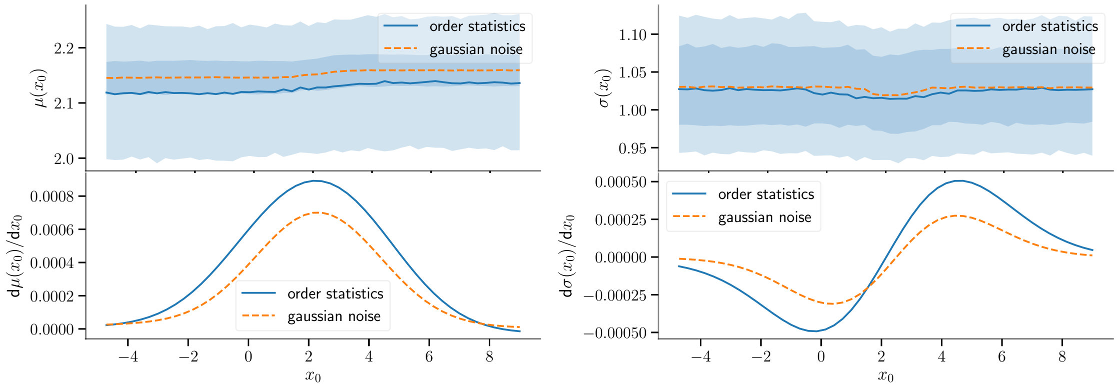


Figure 10.4.3: Change in the posterior (top row) and posterior mean (bottom row) by varying one of N samples.

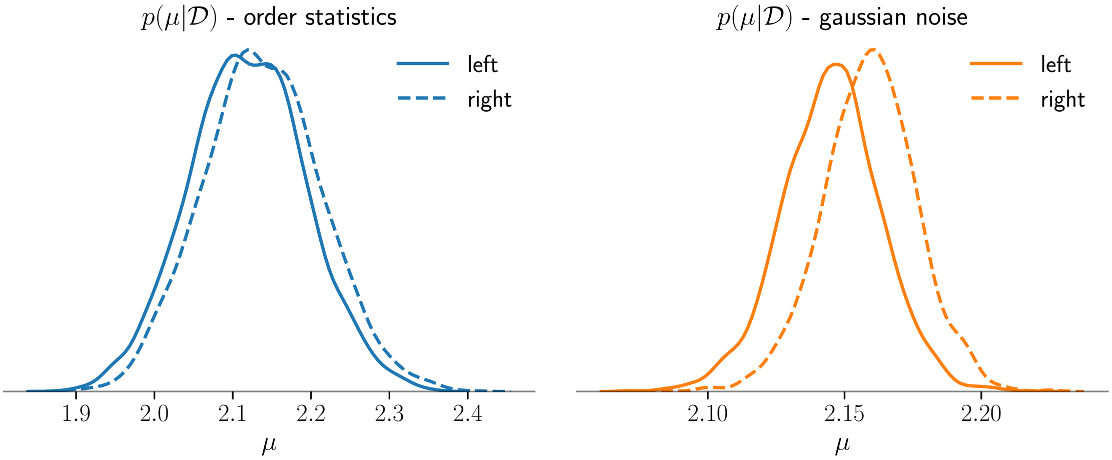


Figure 10.4.4: Change of the posterior distribution by changing one sample out of N.

set to -5 and the dashed line shows the posterior for the fit, where the first element of the sample is set to 9. The left plot shows the change in the posterior $p(\mu|\mathcal{D})$ for the order statistics and the right plot for the Gaussian noise model. This large relative change in the Gaussian noise model case is an indication that the model either is not robust enough or severely underestimates the parameter uncertainty. Note, that we are only changing one out of 200 samples, so we would expect the fit not to change much.

The bottom row of Figure 10.4.3 shows the change of the mean of the posterior. Before calculating the change, we apply a kernel smoother with a Gaussian kernel to get rid of the sampling noise. Here, we see that the posterior of the order statistics model changes earlier (further in the tails) than the posterior of the Gaussian noise model. This again emphasizes the focus on the tails by the order statistics.

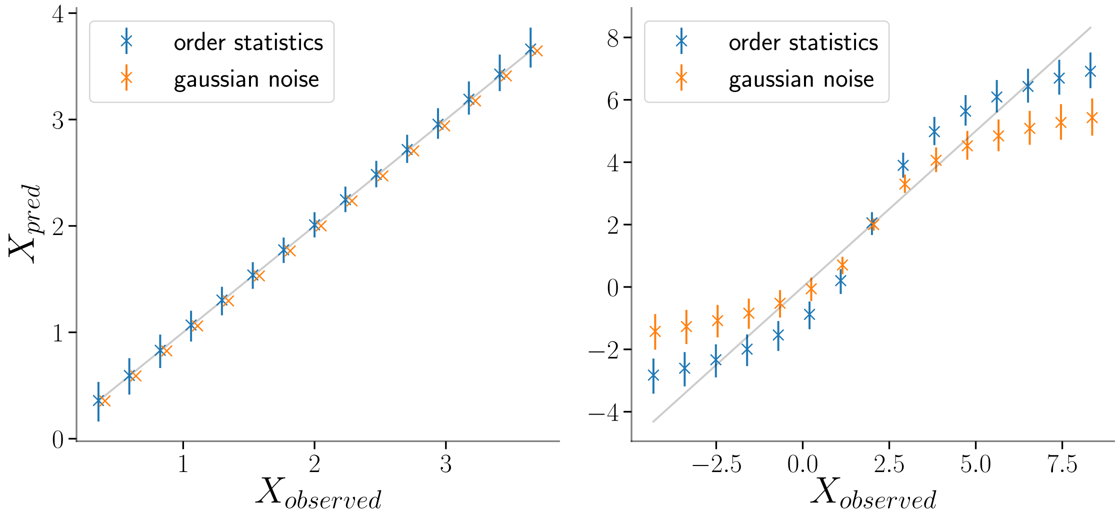


Figure 10.4.5: Fit of a specified (left) and misspecified model (right).

10.4.4 Penalty for OS and CDF-fit

In this section we look into misspecified models. These are models that cannot reproduce the true data generating process for any parameter setting. For example, when the data are coming from a Cauchy distribution and we try to fit them by a Gaussian. The Cauchy distribution has a totally different tail behaviour, namely heavy tails, and a Gaussian has light tail, which fall off double exponentially. Thus a Gaussian can never (for no parameter combination) fit a Cauchy perfectly.

By fitting a misspecified model and comparing the quantile values of the fitted distribution to the true quantile values, we can get more information about the nature of the order statistics and can better compare it with the Gaussian noise model. We already mentioned in Section 10.3 that the penalty of the observed data $(\boldsymbol{q}, \boldsymbol{x}, N)$ is quite different from the order statistics model compared to the Gaussian noise model. This will get quite obvious in this section.

To illustrate that, we take N samples from the data generating distribution, in this case a Cauchy distribution, then calculate M quantile values $\boldsymbol{x}$ for given quantiles $\boldsymbol{q}$. Subsequently, we fit another distribution, in this case a Gaussian distribution to the data $(\boldsymbol{q}, \boldsymbol{x}, N)$ via order statistics and Gaussian noise model.

Figure 10.4.5 shows the fits for a specified model (left) and a misspecified model (right). In each case we took $N = 200$ and 15 equidistant quantile values in the range from 5% to 95%. For the left plot, we took data from a Gaussian distribution with mean 2.0 and variance 1.0 and also fit a Gaussian. The predicted quantile values are totally in line with the true quantile values from a Gaussian. On the right plot, we took data from a Cauchy with location and scale parameters set to 2.0 and 1.0 and fitted a Gaussian distribution. This time the predicted quantile values are not in line with the true quantile values from a Cauchy distribution. Note that each of the model focuses on different parts of the data.

While the Gaussian noise model predicted better (with smaller error) the center of the distribution, the order statistics model predicted better the tails of the distribution. The Gaussian noise model, which correspond to the MSE, does not put much emphasis on the tails. This is due to the fact, that in the tails the difference between the CDF of a heavy tailed distribution (as the Cauchy) and a very light tailed distribution (as the Gaussian) becomes negligible compared to their differences around the mode. Therefore, the contribution of the tails to the overall MSE is very small and leads to the neglect of the tails. The order statistics on the other hand, can get more information from the tails. This is shown in Figure 10.4.6. We plotted the likelihood of the order statistics (blue) and the Gaussian noise model (orange) for a Gaussian with PDF $f(x) = \mathcal{N}(x|2.0, 1.0)$ (green) and CDF F. In particular, we plotted the unnormalized likelihoods for a single points

$$p_{\text{os}}(x|q) \propto F(x)^{qN-1}(1 - F(x))^{N-qN} f(x) \tag{10.4.1}$$

$$p_{\text{gn}}(q|x) \propto \exp\left\{-\frac{1}{2\sigma^2_{\text{noise}}}(F(x) - q)^2\right\} \tag{10.4.2}$$

as a function of x for a fixed quantile $q = 0.01$. p_{os} is the likelihood for the order statistics and p_{gn} is the likelihood for the Gaussian noise model. The black vertical line shows the quantile value x_{true} for the fixed quantile q_{true} [2] and the blue and orange lines show the score that an estimated value x corresponding to a q would get. The order statistics shows a peak at the true value x_{true} and decreases on both sides even if x_{true} is set further in the tails ($q = 0.1, 0.01$). The Gaussian noise model on the other hand does not decrease to zero even far away from the true quantile value. The reason for that is, that the likelihood (Equation 10.4.2, specifically $(F(x) - q)^2$) does not change much for a large change in x. This is the case in particular in the tails. Therefore, e.g.: a bad estimate of $q = F(x) = 0.0001$ for $q_{\text{true}} = 0.01$ leads to almost the same likelihood score as $q = F(x) = 0.01 = q_{\text{true}}$ (Equation 10.4.2). Because of that, the CDF regression with Gaussian noise is totally not suitable for applications where tails of the distribution are important. Order statistics is a better choice here.

10.5 Matching salary data of different countries

In this section we are applying our model to a real world data set. Table 10.1 shows the quantile values for $\boldsymbol{q} = (0.25, 0.50, 0.75)$ of the salary distribution of some European countries. The corresponding sample size is also provided in Table 10.1[3].

The goal is not just to fit the data by some distribution but also select the best model as explained in Section 10.2.3. Therefore, we evaluate the likelihood at the posterior parameter samples and take the model that maximizes the mean of the log-likelihood values.

[2] ppf in the legend of Figure 10.4.6 denotes the percent point function, which is the inverse of the cumulative density function.

[3] The data are downloaded from the Eurostat homepage (Distribution of income by quantiles - EU-SILC and ECHP surveys) at https://ec.europa.eu/eurostat/. Information about the sample size is also available on the website (EU and national quality reports).

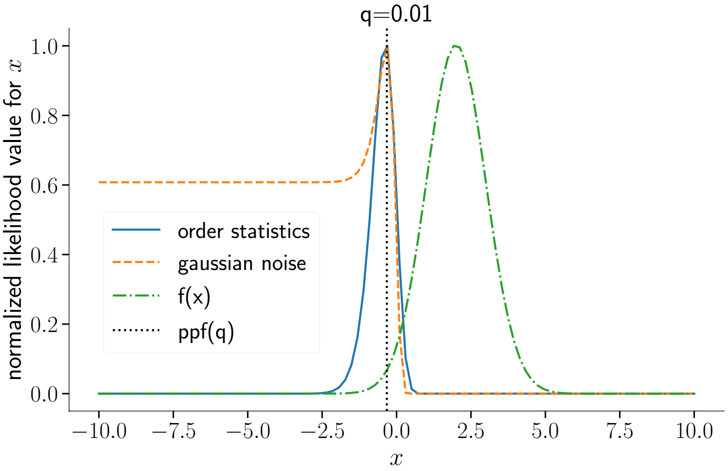


Figure 10.4.6: Emphasis on the tails from Gaussian noise model and order statistics.

Table 10.1: Aggregated salary data of some European countries from 2016.

Country	Sample Size	25	50	75
EL	12918	4930	7500	11000
ES	19177	8803	13681	20413
FR	21325	16185	21713	29008
IT	24969	10699	16247	22944
LU	10292	23964	33818	48692
NL	12748	16879	22733	30327
SE	11635	17794	25164	33365
UK	17645	14897	21136	30151

Before fitting the data, we normalize it by dividing the values of each country by its median (50% quantile). After normalization, we fit the data with various distributions f. For each country, the mean of the log-likelihood samples for 6 different distributions is shown in Table 10.2. The highest log-likelihood values are emphasised in bold writing. The best fits to the quantile salary data are given by the Weibull, Log-normal and Gamma distribution. Bandourian et al. (2002) suggests to use a Weibull distribution as a two parameter distribution for salary data. However, the data they used were not as aggregated as ours. In our case we only have three quantiles available and Table 10.2 shows that the Weibull as well as the Log-normal and the Gamma distributions are reasonable to fit the data.

Figure 10.5.1 shows the posterior predictive cumulative distribution for 10 countries for

Table 10.2: The mean of the log likelihood samples for different models and countries. The + and - values show the distance to the 95 and 5 % quantile of the log-likelihood distribution.

COUNTRY	WEIBULL	LOGNORMAL	GAMMA	INV_GAMMA	FRECHET	CHI_SQUARE
EL	$-6.9^{+0.9}_{-2.1}$	$4.3^{+0.9}_{-2.0}$	$\mathbf{10.2}^{+1.0}_{-2.0}$	$-31.5^{+0.9}_{-2.0}$	$-81.1^{+1.0}_{-2.2}$	$-2063.9^{+0.5}_{-1.3}$
ES	$-13.4^{+1.0}_{-1.9}$	$-0.2^{+1.0}_{-2.1}$	$\mathbf{10.1}^{+0.9}_{-1.8}$	$-58.9^{+1.0}_{-2.0}$	$-130.4^{+1.0}_{-2.1}$	$-2776.2^{+0.5}_{-1.4}$
FR	$-57.7^{+1.0}_{-2.0}$	$\mathbf{13.0}^{+0.9}_{-2.0}$	$3.5^{+1.0}_{-2.2}$	$-4.4^{+1.0}_{-2.1}$	$-76.8^{+0.9}_{-2.0}$	$-5847.6^{+0.5}_{-1.4}$
IT	$\mathbf{9.1}^{+0.9}_{-2.0}$	$-48.9^{+0.9}_{-2.0}$	$5.3^{+0.9}_{-2.0}$	$-155.2^{+0.9}_{-2.0}$	$-290.0^{+1.0}_{-2.1}$	$-4500.3^{+0.5}_{-1.4}$
LU	$-47.7^{+1.0}_{-2.0}$	$\mathbf{9.3}^{+1.0}_{-2.1}$	$-9.3^{+0.9}_{-2.1}$	$9.1^{+0.9}_{-2.0}$	$-10.9^{+0.9}_{-1.9}$	$-2062.3^{+0.5}_{-1.4}$
NL	$-23.3^{+0.9}_{-1.9}$	$\mathbf{11.6}^{+1.0}_{-2.0}$	$9.0^{+0.9}_{-1.9}$	$-1.9^{+1.0}_{-2.0}$	$-49.4^{+1.0}_{-2.1}$	$-3473.8^{+0.5}_{-1.4}$
SE	$\mathbf{11.4}^{+1.0}_{-2.0}$	$-21.2^{+0.9}_{-2.0}$	$3.9^{+1.0}_{-2.0}$	$-63.0^{+1.0}_{-2.0}$	$-138.7^{+0.9}_{-2.0}$	$-2910.8^{+0.5}_{-1.3}$
UK	$-62.8^{+0.9}_{-2.0}$	$\mathbf{12.1}^{+1.0}_{-2.0}$	$-8.1^{+1.0}_{-2.0}$	$0.5^{+0.9}_{-1.9}$	$-45.6^{+0.9}_{-1.9}$	$-3582.7^{+0.5}_{-1.3}$

the best fit. The posterior predictive cumulative distribution is given by

$$P(X < x') = \int F_{\boldsymbol{\theta}}(x') p(\boldsymbol{\theta}|\boldsymbol{x}, \boldsymbol{q}, N) \mathrm{d}\boldsymbol{\theta} \, . \tag{10.5.1}$$

This distribution also allows us to do more than the original three quantile values. Questions like the following might be asked: What is the threshold for the 99% quantile (earnings of the top 1%)? Table 10.3 shows estimates of the 99% quantiles for the best fits for each country. To detect misspecification, we can compare the true and the predicted quantile values. Figure 10.5.2 compares the predictive distribution of different models to the observed quantiles for UK. According to Table 10.2, Log-normal is the best distribution and this is also visually clear in Figure 10.5.2, where being only based on three quantile, the Log-normal shows no sign of misspecification in contrast to the other distributions.

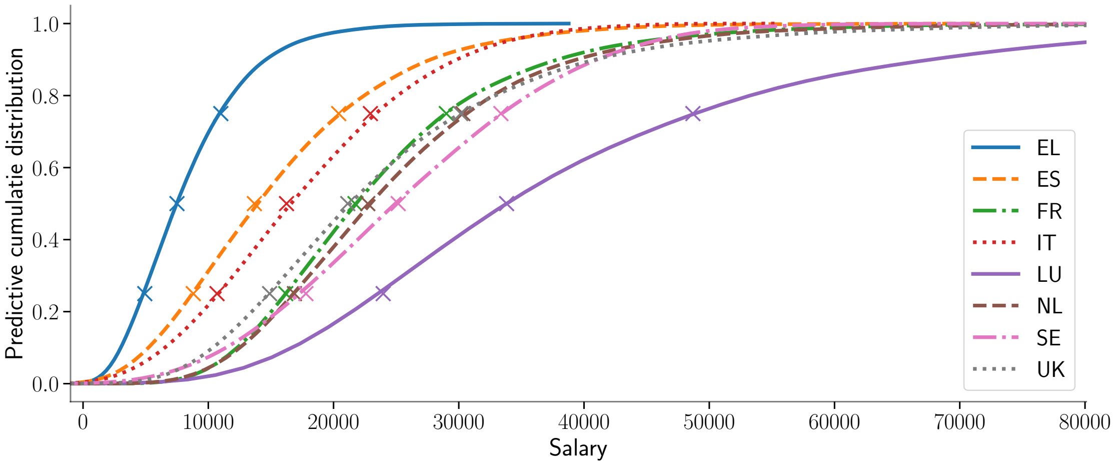


Figure 10.5.1: Posterior predictive cumulative distribution for observed data marked by the cross.

Table 10.3: The earnings of the top 1% according to the best model based on the log-likelihood score. The + and - values show the distance to the 95 and 5 % quantile of the distribution.

Country	best model	99% quantile
EL	gamma	$23268.6\ ^{+406.8}_{-400.4}$
ES	gamma	$44343.5\ ^{+701.7}_{-633.7}$
FR	lognormal	$59331.9\ ^{+834.6}_{-838.8}$
IT	weibull	$41096.2\ ^{+483.8}_{-467.6}$
LU	lognormal	$115693.5\ ^{+3038.5}_{-2796.1}$
NL	lognormal	$62265.1\ ^{+1185.3}_{-1142.6}$
SE	weibull	$53926.5\ ^{+754.0}_{-744.5}$
UK	lognormal	$71466.4\ ^{+1294.2}_{-1280.7}$

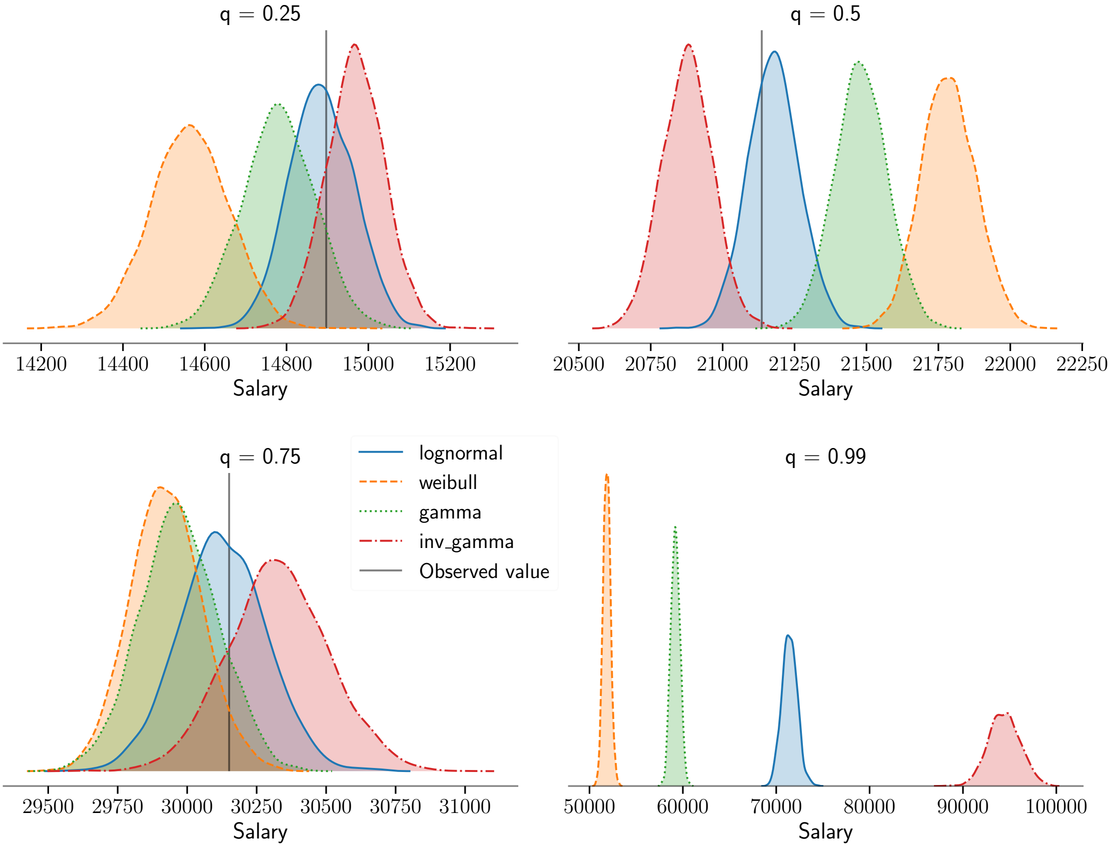


Figure 10.5.2: The top row and the left side of the bottom row shows the predictions made for the quantile that we also observed. The bottom right plot shows an out of sample prediction for the 99% quantile. Different colors indicate different models. The best model according to the log-likelihood value is the Log-normal distribution. This figure visually validates that result.

Chapter 11

Summary and conclusion

In this thesis, we provide a more extended version of the published work on the Gaussian process latent variable models in finance (Nirwan and Bertschinger, 2019b), rotation invariant Householder parameterization of Bayesian PCA (Nirwan and Bertschinger, 2019a) and Bayesian quantile matching estimation (Nirwan and Bertschinger, 2020). The extension includes a more detailed introduction to the topics, description of the models and experiments.

The focus of the thesis is on the use of probabilistic models (specially Gaussian processes and Gaussian process latent variable models) in finance. Chapter 7 provides a detailed description on how that was done. This work is also published in Nirwan and Bertschinger (2019b). While using GPLVMs, we encountered a problem of rotation invariance latent space and solved it by reparameterizing the model. The procedure is described in Chapter 8 and is published in Nirwan and Bertschinger (2019a). Chapter 10 includes work on Bayesian quantile matching estimation, where we suggest a new method for matching quantiles to a distribution by using order statistics. Our method can use aggregated data to learn a distribution. We used that method to fit salary data of some European countries which are only available in an aggregated form. This work is not published yet but a preprint is available on arxiv (Nirwan and Bertschinger, 2020).

After the Introduction in Chapter 1, we start with the basics on Bayesian modeling and explain the Bayesian framework with an example of the linear regression in Chapter 2. More complex models are analytically not tractable anymore and we need to resort to approximation methods. The approximation methods that are used in this thesis (HMC and VI) are described in Chapter 3. Financial modeling is done using Gaussian processes (GPs) and Gaussian process latent variable models (GPLVMs). Thus Chapter 4 and Chapter 5 introduce GPs and GPLVMs, respectively. In Chapter 6, we start by introducing some basic financial models and extend them to GPLVMs in Chapter 7. Here, we conduct experiments, where we fit financial data with GPLVMs. At the end, we introduce some financial applications, where the GPLVM and the GPs can be applied. The GPLVM has a rotation invariancy in the latent space, which makes the model harder to fit. Chapter 8 deals with the problem of rotation invariant latent spaces and suggests a new parameterization based on Householder transformations that breaks the rotational symmetry. The final two chapters are about matching empirical quantiles to a distribution. Chapter 9 introduces the theory behind the proposed method, which is the order statistics of sample values. Chapter 10 explains the proposed method. We also compare the performance of the suggested method to existing methods and conduct experiments on real world data.

11.1 Gaussian process latent variable models in finance

The first topic was about the application of Gaussian process latent variable model (GPLVM) in Finance. We extended the multi-index model (APT - Arbitrage theory of capital asset pricing) and showed that its nonlinear form can be expressed by a GPLVM. GPLVM reduces to APT when using the linear kernel. Additionally, we used the GPLVM to analyse financial data in many other ways. The structure of the data that was captured by the model, measured in the R^2-score, increased with switching from linear to nonlinear kernels for fixed latent space dimension Q. However, due to the nonlinearity we lose the interpretability of the columns of the latent space matrix $\boldsymbol{X}$ as latent factors. We applied the GPLVM to the return matrix of stocks and estimated the covariance matrix $\boldsymbol{K}$ and an efficient latent space representation $\boldsymbol{X}$ of them. $\boldsymbol{K}$ and $\boldsymbol{X}$ can then be used for many other applications, among which we illustrated three cases.

Portfolio allocation: First of all, we used the covariance of different stocks $\boldsymbol{K}$ to allocate a minimal risk portfolio according to the modern portfolio theory. Here, we backtested the resulting portfolio on the S&P500 from 2002 to 2018 and compared the risk of the resulting portfolio to the risk of other portfolios (Sample covariance, Ledoit Wolf estimate of the covariance and the equally weighted portfolio). The estimates by the GPLVM for nonlinear kernels had the minimal risk.

Prediction of missing values: Using the GPLVM, we constructed the latent space representation of the stocks $\boldsymbol{X}$. The latent space representation allowed us to fill in missing values of the return of a stock given the observation of other stocks at that day. This was done using a standard GP regression, which provided us the conditional probability of the missing return given the observed returns of other assets on a particular day.

Latent space representation: For stationary kernels, the distances between the stocks in the latent space are directly related to their correlation. We showed how this can lead to a clustering of stocks into their subsectors. The results can be applied for structure detection in financial data. It can also be used to construct portfolios with decorrelated assets. For example, one can build a portfolio of M assets, which are chosen such that their sum of distances is maximized. Another option is to simply buy the convex hull.

11.2 Rotation invariant Householder parameterization for Bayesian PCA

The problem with the GPLVM is that if one is interested in the latent space, the posterior draws are not helpful because of the rotational symmetry. To deal with that problem, we proposed another parameterization for the model. The new parameterization does not change the model (i.e. the data distribution is still the same) but breaks the rotational symmetry of the latent space. Therefore, we first started with the probabilistic PCA (GPLVM with a linear kernel) and solved the rotation invariancy problem there. The idea was to parameterize the model by the singular value decompositions $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ of $\boldsymbol{W}$ instead of $\boldsymbol{W}$ directly. By setting the redundant rotation of the latent space $\boldsymbol{V}$ to identity, we were able to break the rotational symmetry of the posterior. We identified the right prior on $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ such that the probability in the data space (i.e. the model) is not changed. The benefits of using the new parameterization are discussed as follows.

Less parameters: The new parametrization uses Householder reflections to parameterize the orthogonal matrix $\boldsymbol{U}$ and has the same degrees of freedom as orthogonal matrices, namely $DQ - \frac{1}{2}Q(Q+1)$. Other Q dimensions are coming from the Q principle variances in $\boldsymbol{\Sigma}$. In contrast, the parameterization by $\boldsymbol{W}$ has DQ parameters. In the proposed algorithm, however, we increased the number of parameters by Q. This was done to simplify the sampling from a sphere. The extra dimensions allow the sampler to smoothly move in the parameter space.

Computational efficiency: The known distribution on the $\boldsymbol{v}$s, which we used to parameterize $\boldsymbol{U}$, to make sure the resulting matrix is Haar distributed, saved a lot of computation time. The efficiency gain is the result of not needing a Jacobian correction for the transformation. Other parameterizations, e.g. givens rotations (Shepard et al., 2014), need a Jacobian correction for $\mathcal{O}(DQ)$ parameters that has $\mathcal{O}(D^3Q^3)$ complexity.

Interpretability: Another huge benefit of the new parameterization is the interpretability of $\boldsymbol{U}$ as data space rotation and $\boldsymbol{\Sigma}$ as principle variance. This allows us to decode our a-priori knowledge into the prior for both, the rotation and the principle variances.

Easy extension to nonlinear models: At the end, we also successfully applied the results to nonlinear model. In particular, we tested the new parameterization for the GPLVM with nonlinear stationary kernels and broke the latent space symmetry there as well.

11.3 Bayesian quantile matching estimation

The last project was about the Bayesian matching of empirical quantiles to a distribution by using order statistics. Here, we suggested an alternative approach for quantile matching estimation to the widely used MSE (mean squared error) minimization. The MSE minimization is equivalent to the likelihood maximization with the likelihood being a Gaussian. After the review of the theory of order statistics, we proposed a method that allows the use of order statistics to match empirical quantiles to a distribution. There are a number of advantages of this method compared to the Gaussian noise model, which are discussed below.

Capturing the tails of a distribution: Our model correctly accounts for the higher uncertainty of the tail quantiles, whereas the Gaussian noise model overemphasises the central part of the distribution and neglects the tails.

Estimation of parameter uncertainty: The Bayesian approach allows for better assessment of model uncertainty. Our experiments on synthetic data sets with known parameters validate, that the Gaussian noise model (also the Bayesian variant) substantially underestimates the true uncertainty in the parameter estimates, whereas our model accurately reflects that uncertainty.

Model selection: We also showed, how model selection can be performed with our proposed method. Specifically, we showed how to select the right modeling distribution when several candidates are given.

We empirically showed all the benefits listed above and at the end, we applied our approach to fit salary data of several European countries. Among several distributions the Weibull, Log-normal and the Gamma distribution fitted the data the best.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Ambrogioni, L., Güçlü, U., Güçlütürk, Y., Hinne, M., van Gerven, M. A. J., and Maris, E. (2018). Wasserstein variational inference. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 2473–2482. Curran Associates, Inc.

Awoyemi, J. O., Adetunmbi, A. O., and Oluwadare, S. A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 International Conference on Computing Networking and Informatics (ICCNI)*, pages 1–9.

Bai, Z. D., Miao, B. Q., and Pan, G. M. (2007). On asymptotics of eigenvectors of large sample covariance matrix. *Ann. Probab.*, 35(4):1532–1572.

Bandourian, R., Turley, R., and McDonald, J. (2002). A Comparison of Parametric Models of Income Distribution across Countries and over Time. LIS Working papers 305, LIS Cross-National Data Center in Luxembourg.

Bao, W., Lianju, N., and Yue, K. (2019). Integration of unsupervised and supervised machine learning algorithms for credit risk assessment. *Expert Systems with Applications*, 128:301 – 315.

Bassett, D. and Sporns, O. (2017). Network neuroscience. *Nature Neuroscience*, 20:353–364.

Bayarri, M. J. and Berger, J. O. (2004). The interplay of bayesian and frequentist analysis. *Statist. Sci.*, 19(1):58–80.

Belgorodski, N., Greiner, M., Tolksdorf, K., Schueller, K., Flor, M., and Göhring, L. (2017). rriskDistributions: Fitting distributions to given data or known quantiles.

Besag, J., Green, P., Higdon, D., and Mengersen, K. (1995). Bayesian computation and stochastic systems. *Statistical Science*, 10(1):3–41.

Betancourt, M. (2017). A conceptual introduction to hamiltonian monte carlo. arxiv:1701.02434.

Betancourt, M. J. (2014). Adiabatic monte carlo.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Burt, D., Rasmussen, C. E., and Van Der Wilk, M. (2019). Rates of convergence for sparse variational Gaussian process regression. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871, Long Beach, California, USA. PMLR.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32.

Chai, L. R., Khambhati, A. N., Ciric, R., Moore, T. M., Gur, R. C., Gur, R. E., Satterthwaite, T. D., and Bassett, D. S. (2017). Evolution of brain network dynamics in neurodevelopment. *Network Neuroscience*, 1(1):14–30.

Clyde, M. and Iversen, E. (2013). *Bayesian Model Averaging in the M-Open Framework*, pages 484–498.

Dominicy, Y. and Veredas, D. (2013). The method of simulated quantiles. *Journal of Econometrics*, 172(2):235 – 247. Latest Developments on Heavy-Tailed Distributions.

Everett, B. (1984). *An Introduction to Latent Variable Models*. Chapman and Hall.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417.

Fama, E. F. and French, K. R. (1993). Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics*, 33:3–56.

Fama, E. F. and French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22.

Financial Accounting Standards Board (2006). Statement of financial accounting standards no. 157 – fair value measurements. Norwalk, CT.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459. On Probabilistic models.

Gikhman, I. I. and Skorokhod, A. V. (1974). *The theory of stochastic processes / I. I. Gihman, A. V. Skorohod ; translated from the Russian by S. Kotz.* Springer-Verlag Berlin ; New York.

Golub, G. H. and van Loan, C. F. (2013). *Matrix Computations.* JHU Press, fourth edition.

Henry, K., Hager, D., Pronovost, P., and Saria, S. (2015). A targeted real-time early warning score (trewscore) for septic shock. *Science translational medicine*, 7:299ra122.

Hoffman, M., Sountsov, P., Dillon, J. V., Langmore, I., Tran, D., and Vasudevan, S. (2019). Neutra-lizing bad geometry in hamiltonian monte carlo using neural transport.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(47):1593–1623.

James, O. and Lee, H.-N. (2014). Concise Probability Distributions of Eigenvalues of Real-Valued Wishart Matrices. *ArXiv e-prints.*

Jefferys, W. H. and Berger, J. O. (1992). Ockham's Razor and Bayesian Analysis. *American Scientist*, 80(1):64–72.

Jobson, J. D. and Korkie, R. M. (1981). Putting markowitz theory to work. *The Journal of Portfolio Management*, 7(4):70–74.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.*, 32(11):1238–1274.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. (2017). Automatic differentiation variational inference. *J. Mach. Learn. Res.*, 18(1):430–474.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816.

Ledoit, O. and Wolf, M. (2004). Honey, I shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119.

Li, Y. and Turner, R. E. (2016). Rényi divergence variational inference. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 1073–1081. Curran Associates, Inc.

MacKay, D. J. C. (2002). *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, USA.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

Mezzadri, F. (2007). How to generate random matrices from the classical compact groups. *Notices of the American Mathematical Society*, 54(5):592 – 604.

Minka, T. P. (2001). Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, page 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Neal, R. M. (2010). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162.

Nirwan, R. and Bertschinger, N. (2019a). Rotation invariant householder parameterization for Bayesian PCA. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4820–4828, Long Beach, California, USA. PMLR.

Nirwan, R. S. and Bertschinger, N. (2019b). Applications of gaussian process latent variable models in finance. In Bi, Y., Bhatia, R., and Kapoor, S., editors, *Intelligent Systems and Applications*, pages 1209–1221, Cham. Springer International Publishing.

Nirwan, R.-S. and Bertschinger, N. (2020). Bayesian quantile matching estimation.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pourzanjani, A. A., Jiang, R. M., Mitchell, B., Atzberger, P. J., and Petzold, L. R. (2017). General Bayesian Inference over the Stiefel Manifold via the Givens Transform. *ArXiv e-prints*.

Rasmussen, C. E. and Ghahramani, Z. (2001). Occam's razor. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, page 276–282, Cambridge, MA, USA. MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Ross, S. A. (1976). The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3):341 – 360.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. cite arxiv:1609.04747.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319.

Sgouropoulos, N., Yao, Q., and Yastremiz, C. (2015). Matching a distribution by matching quantiles estimation. *Journal of the American Statistical Association*, 110(510):742–759. PMID: 26692592.

Sharpe, W. F. (1964). Capital Asset Prices: A Theory Of Market Equilibrium Under Conditions Of Risk. *Journal of Finance*, 19(3):425–442.

Sharpe, W. F. (1966). Mutual fund performance. *The Journal of Business*, 39(1):119–138.

Shepard, R., Gidofalvi, G., and Brozell, S. R. (2014). The multifacet graphically contracted function method. II. a general procedure for the parameterization of orthogonal matrices and its application to arc factors. *The Journal of Chemical Physics*, 141(6):064106.

Snelson, E. and Ghahramani, Z. (2006). Variable noise and dimensionality reduction for sparse gaussian processes. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 461–468, Arlington, Virginia, USA. AUAI Press.

Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 61(3):611–622.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In van Dyk, D. and Welling, M., editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.

Uhlig, H. (1994). On singular wishart and singular multivariate beta distributions. *Ann. Statist.*, 22(1):395–405.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Wilson, A. G. and Adams, R. P. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *ICML (3)*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1067–1075. JMLR.org.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. (2019). Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026.

Appendix A

Appendix

A.1 Normalization Constant

To verify that Equation (9.3.5) is indeed the normalization constant, we have to integrate Equation (9.3.4) with respect to all u's

$$\frac{1}{c} = \int_0^{u_2} \int_{u_1}^{u_3} \cdots \int_{u_{M-1}}^{1} u_1^{k_1-1}(1-u_M)^{n-k_M} \prod_{m=2}^{M} (u_m - u_{m-1})^{k_m-k_{m-1}-1} \mathrm{d}u_1 \mathrm{d}u_2 \ldots \mathrm{d}u_M. \tag{A.1.1}$$

For the integration of a particular u_i, however, only the following term is relevant

$$\int_{u_{i-1}}^{u_{i+1}} (u_i - u_{i-1})^{k_i-k_{i-1}-1} (u_{i+1} - u_i)^{k_{i+1}-k_i-1} \mathrm{d}u_i. \tag{A.1.2}$$

The rest is constant with respect to u_i. To solve this integral we substitute $u = \frac{u_i - u_{i-1}}{u_{i+1} - u_{i-1}}$ for u_i and get

$$\begin{aligned} &\int_{u_{i-1}}^{u_{i+1}} (u_i - u_{i-1})^{k_i-k_{i-1}-1} (u_{i+1} - u_i)^{k_{i+1}-k_i-1} \mathrm{d}u_i \\ &\qquad = (u_{i+1} - u_{i-1})^{k_{i+1}-k_{i-1}-1} \int_0^1 u^{k_i-k_{i-1}-1} (1-u)^{k_{i+1}-k_i-1} du \\ &\qquad = (u_{i+1} - u_{i-1})^{k_{i+1}-k_{i-1}-1} \frac{\Gamma(k_i - k_{i-1})\Gamma(k_{i+1} - k_i)}{\Gamma(k_{i+1} - k_{i-1})}. \end{aligned} \tag{A.1.3}$$

Note that the integrand in the second step is a unnormalized beta distribution, where we already know the normalization constant (Bishop, 2006). $\Gamma(\cdot)$ is the Gamma-function, which, for integer input has the form $\Gamma(n+1) = n!$.

By integrating out u_i, the resulting expression has still a similar form. Thus, by successive applications of the above result, we obtain the normalization constant as in Equation (9.3.5).