Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations

Matthias Seeger

Doctor of Philosophy

Institute for Adaptive and Neural Computation
Division of Informatics
University of Edinburgh

2003

Abstract

Non-parametric models and techniques enjoy a growing popularity in the field of machine learning, and among these Bayesian inference for Gaussian process (GP) models has recently received significant attention. We feel that GP priors should be part of the standard toolbox for constructing models relevant to machine learning in the same way as parametric linear models are, and the results in this thesis help to remove some obstacles on the way towards this goal.

In the first main chapter, we provide a distribution-free finite sample bound on the difference between generalisation and empirical (training) error for GP classification methods. While the general theorem (the PAC-Bayesian bound) is not new, we give a much simplified and somewhat generalised derivation and point out the underlying core technique (convex duality) explicitly. Furthermore, the application to GP models is novel (to our knowledge). A central feature of this bound is that its quality depends crucially on task knowledge being encoded faithfully in the model and prior distributions, so there is a mutual benefit between a sharp theoretical guarantee and empirically well-established statistical practices. Extensive simulations on real-world classification tasks indicate an impressive tightness of the bound, in spite of the fact that many previous bounds for related kernel machines fail to give non-trivial guarantees in this practically relevant regime.

In the second main chapter, *sparse* approximations are developed to address the problem of the unfavourable scaling of most GP techniques with large training sets. Due to its high importance in practice, this problem has received a lot of attention recently. We demonstrate the tractability and usefulness of simple greedy forward selection with information-theoretic criteria previously used in active learning (or sequential design) and develop generic schemes for automatic model selection with many (hyper)parameters. We suggest two new generic schemes and evaluate some of their variants on large real-world classification and regression tasks. These schemes and their underlying principles (which are clearly stated and analysed) can be applied to obtain sparse approximations for a wide regime of GP models far beyond the special cases we studied here.

Acknowledgements

During the course of my PhD studies I have been fortunate enough to benefit from inspiring interactions with people in the machine learning community, and I am delighted to be able to acknowledge these here.

My thanks go first of all to my supervisor Christopher Williams who guided me through my research, and whose comments often sparked my deeper interest in directions which I would have ignored otherwise, or forced me to re-consider intuitive argumentations more carefully. I have also benefitted from his wide overview of relevant literature in many areas of machine learning I have been interested in.

I would like to thank the members of the ANC group in Edinburgh for many inspiring discussions and interesting talks covering a wide range of topics. Especially, I would like to thank Amos Storkey and David Barber for having shared their knowledge and pointing me to interesting work in areas beyond the necessarily narrow scope of this thesis. I gratefully acknowledge support through a research studentship from Microsoft Research Ltd.

The "random walk" of my postgraduate studies led me into terrain which was quite unfamiliar to me, and it would have been a much harder and certainly much more boring journey without some people I met on the way. Ralf Herbrich shared some of his deep and pragmatic insights into data-dependent distribution-free bounds and other areas of learning theory and gave many helpful comments, and I very much enjoyed the time I spent working with him and Hugo Zaragoza during an internship at MS Research, Cambridge in the autumn of 2000, not least the late-night pub sessions in the Eagle. I am grateful for many discussions with Neil Lawrence, Michael Tipping, Bernhard Schölkopf and Antonio Criminisi who were there at that time.

I was fortunate enough to work with John Langford, one of whose diverse interests in learning theory are applications and refinements of the PAC-Bayesian theorem. I have learned a lot from his pragmatic approach to learning-theoretical problems, and I would enjoy to do further joint work with him in the future. I am very grateful to Manfred Opper for sharing some of his enormous knowledge about

learning-theoretical analyses of Bayesian procedures. Manfred got interested in my work, parts of which reminded him of studies of learning curves for Bayesian methods he had done some time ago with David Haussler, and invited me to Aston University, Birmingham for a few days. Our discussions there were invaluable in that they helped me to abstract from the details and see the big picture behind the PAC-Bayesian technique, recognising for the first time the huge versatility of convex inequalities. I would also like to thank David McAllester for proposing and proving the remarkable PAC-Bayesian theorems in the first place, and for some very interesting discussions when we met at Windsor, UK and at NIPS.

My interest in sparse Gaussian process approximations was sparked by work with Chris Williams (see Section 4.7.1) but also by my frustration with running time and memory consumption for my experiments with the PAC-Bayesian theorem. My goal was to provide a PAC result which is practically useful, but of course this calls for a *method* which practitioners can really use on large datasets. I got interested in the *informative vector machine (IVM)* presented by Neil Lawrence and Ralf Herbrich in a NIPS workshop contribution and worked out some generalisations and details such as the expectation propagation (EP) foundation and the randomisation strategy, so that my implementation would be able to handle datasets of the size required for the PAC-Bayesian experiments. I enjoyed this collaboration (which led to more joint work with Neil Lawrence).

I would like to thank John Platt and Christopher Burges for giving me the opportunity to spend the summer of 2002 doing research at Microsoft, Redmond. I enjoyed the work with Chris and would have liked to spend more time with John who, unfortunately for me but very fortunately for him, became father at that time. I would also like to thank Patrice Simard for some interesting discussions and for many great football ("soccer") matches (in the end "youth and stamina" prevailed once more over "experience and wisdom"). Thanks also to Alex Salcianu for joining me on some great hiking trips around Mt. Rainier, fishing me out of Lake Washington after some unexpected canoeing manoeuvre, and for his persistence of getting the photos of our mind-boggling white water rafting day. I hope we meet again for some hiking, although then I would prefer

to do the driving!

Finally, I would like to thank the friends I have made during my time in this beautiful town of Edinburgh. During my first year, Esben and I explored many of the historical sites in this fascinating country, and he gave me a very lively account of truths and myths in Scottish history. I had a great time with Thomas and Björn who are now out to build artificial worms and to save the rain forest. The amazing scenery and remoteness of the Scottish highlands sparked my interest in hiking and mountain walking, and I would like to thank Karsten, Björn, David, Steve and many others for great experiences and views in this vast wilderness. I am indebted to Emanuela for sharing these years with me, and although we are going separate ways now I wish her all the best in the future.

My special thanks go to my mother and sisters who supported me through these years, especially during the final hardest one. This thesis is written in memory of my father.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Matthias Seeger*)

Table of Contents

1 Introduction **1**

1.1 Declaration of Previous Work, Collaborations 6

1.1.1 Publications during Postgraduate Studies 7

2 Background **11**

2.1 Bayesian Gaussian Processes 11

2.1.1 Gaussian Processes: The Process and the Weight Space View 12

2.1.2 Some Gaussian Process Models 20

2.1.3 Approximate Inference and Learning 26

2.1.4 Reproducing Kernel Hilbert Spaces 31

2.1.5 Penalised Likelihood. Spline Smoothing 36

2.1.6 Maximum Entropy Discrimination. Large Margin Classifiers 40

2.1.7 Kriging 45

2.1.8 Choice of Kernel. Kernel Design 48

2.2 Learning Theory 66

2.2.1 Probably Approximately Correct 67

2.2.2 Concentration Inequalities 71

2.2.3 Vapnik-Chervonenkis Theory 73

2.2.4 Using PAC Bounds for Model Selection 76

3 PAC-Bayesian Bounds for Gaussian Process Methods **79**

3.1 Data-dependent PAC Bounds and PAC-Bayesian Theorems 80

3.1.1 The Need for Data-dependent Bounds 80

3.1.2 Bayesian Classifiers. PAC-Bayesian Theorems 84

3.2 The PAC-Bayesian Theorem for Gibbs Classifiers 85
3.2.1 The Binary Classification Case 86
3.2.2 Confusion Distributions and Multiple Classes 88
3.2.3 Comments 93
3.2.4 The Case of General Bounded Loss 96
3.2.5 An Extension to the Bayes Classifier 97
3.2.6 Some Speculative Extensions 99
3.3 Application to Gaussian Process Classification 102
3.3.1 PAC-Bayesian Theorem for GP Classification 102
3.3.2 Laplace Gaussian Process Classification 104
3.3.3 Sparse Greedy Gaussian Process Classification 106
3.3.4 Minimum Relative Entropy Discrimination 107
3.4 Related Work 108
3.4.1 The Theorem of Meir and Zhang 109
3.5 Experiments 114
3.5.1 The Setup MNIST2/3 114
3.5.2 Experiments with Laplace GPC 115
3.5.3 Experiments with Sparse Greedy GPC 117
3.5.4 Comparison with PAC Compression Bound 119
3.5.5 Using the Bounds for Model Selection 123
3.5.6 Comparing Gibbs and Bayes Bounds 127
3.6 Discussion 129

4 Sparse Gaussian Process Methods **131**
4.1 Introduction 132
4.2 Likelihood Approximations and Greedy Selection Criteria 133
4.2.1 Likelihood Approximations 134
4.2.2 Greedy Selection Criteria 136
4.3 Expectation Propagation for Gaussian Process Models 140
4.4 Sparse Gaussian Process Methods: Conditional Inference 143
4.4.1 The Informative Vector Machine 143
4.4.2 Projected Latent Variables 150

4.5 Model Selection . . . 159
4.5.1 Model Selection for PLV . . . 159
4.5.2 Model Selection for IVM . . . 161
4.5.3 Details about Optimisation . . . 161
4.6 Related Work . . . 164
4.7 Addenda . . . 171
4.7.1 Nyström Approximations . . . 171
4.7.2 The Importance of Estimating Predictive Variances . . . 174
4.8 Experiments . . . 179
4.8.1 IVM Classification: Digit Recognition . . . 179
4.8.2 PLV Regression: Robot Arm Dynamics . . . 182
4.8.3 IVM Regression: Robot Arm Dynamics . . . 190
4.8.4 IVM Classification: Digit Recognition II . . . 193
4.9 Discussion . . . 195

5 Conclusions and Future Work **197**
5.1 PAC-Bayesian Bounds for Gaussian Process Methods . . . 197
5.1.1 Suggestions for Future Work . . . 199
5.2 Sparse Gaussian Process Methods . . . 200
5.2.1 Suggestions for Future Work . . . 202

A General Appendix **205**
A.1 Notation . . . 205
A.1.1 Linear Algebra . . . 205
A.1.2 Probability. Miscellaneous . . . 206
A.2 Linear Algebra. Useful Formulae . . . 208
A.2.1 Partitioned Matrix Inverses. Woodbury Formula. Schur Complements . . . 208
A.2.2 Update of Cholesky Decomposition . . . 209
A.2.3 Some Useful Formulae . . . 211
A.3 Convex Functions . . . 211

A.4 Exponential Families. Gaussians 216
A.4.1 Exponential Families 217
A.4.2 I-Projections 220
A.4.3 Gaussian Variables 221
A.5 Pattern Recognition 225
A.6 Bayesian Inference and Approximations 226
A.6.1 Probabilistic Modelling 227
A.6.2 Bayesian Analysis 228
A.6.3 Approximations to Bayesian Inference 229
A.6.4 Lower Bound Maximisation. Expectation Maximisation . . 232
A.7 Large Deviation Inequalities 236

B Appendix for Chapter 3 **239**
B.1 Extended PAC-Bayesian Theorem: an Example 239
B.2 Details of Proof of Theorem 3.2 241
B.3 Proof of Theorem 3.3 242
B.4 Efficient Evaluation of the Laplace GP Gibbs Classifier 244
B.5 Proof of Theorem 3.4 245
B.5.1 The Case of Regression 247
B.6 Proof of a PAC Compression Bound 248
B.6.1 Examples of compression schemes 252

C Appendix for Chapter 4 **255**
C.1 Expectation Propagation 255
C.1.1 Expectation Propagation for Exponential Families 255
C.1.2 ADF Update for some Noise Models 260
C.2 Likelihood Approximations. Selection Criteria 262
C.2.1 Optimal Sparse Likelihood Approximations 262
C.2.2 Relaxed Likelihood Approximations 264
C.2.3 Cheap versus Expensive Selection Criteria 264
C.3 Derivations for the Informative Vector Machine 265
C.3.1 Update of the Representation 265

C.3.2 Exchange Moves 268
C.3.3 Model Selection Criterion and Gradient 270
C.4 Derivations for Projected Latent Variables 273
C.4.1 Site Approximation Updates. Point Inclusions 273
C.4.2 Information Gain Criterion 275
C.4.3 Extended Information Gain Criterion 277
C.4.4 Gradient of Model Selection Criterion 285

Bibliography **291**

List of Figures

2.1 Sample paths Gaussian covariance function 53
2.2 Sample paths Matérn covariance function 55
2.3 Sample paths exponential covariance function 56
2.4 Sample paths polynomial covariance function 57

3.1 Relation between margin and gap bound part 105
3.2 Contribution of pattern to Gibbs error and Bayes margin loss . . 111
3.3 Upper bound values against expected test errors 123
3.4 Upper bound values against expected test errors (regression) . . . 124
3.5 Upper bound values, expected test errors and gap bound values . 125
3.6 Expected training errors against expected test errors (regression) . 126
3.7 Expected test errors against upper bound values on MNIST8/9 . 128

4.1 IVM: Test error against rejection rate on MNIST, 9 vs. rest 177
4.2 IVM: Comparing different criteria to rank predictions (MNIST, 9 vs. rest) 178
4.3 IVM: Predictive std.dev. against predictive mean (MNIST, 9 vs. rest) 178
4.4 Test error against rejection rate for SVM / IVM (MNIST, 9 vs. rest) 181
4.5 Learning curves for sparse regression methods (kin-40k) 185
4.6 Learning curves for sparse regression methods (pumadyn-32nm) . 187
4.7 Model selection criterion for full and sparse GP regression methods 188
4.8 Model selection runs for full and sparse GP regression methods . . 190
4.9 IVM: Test error against rejection rate (USPS, all classes) 195

A.1 Illustrations of convex duality 213

List of Tables

3.1 Experimental results for Laplace GPC 116
3.2 Experimental results for sparse GPC (IVM) 118
3.3 Experimental results for compression bound (IVM) 120
3.4 Experimental results for compression bound (SVM) 121

4.1 Test errors and training times IVM vs. SVM (MNIST, c vs. rest) . 180
4.2 IVM vs. SVM on MNIST tasks (full-size images) 183
4.3 Training times for sparse regression methods 186
4.4 Model selection for full and sparse GP regression methods 189
4.5 IVM: Test error rates and model selection times (USPS, c vs. rest) 194

Chapter 1

Introduction

In order to solve real-world machine learning problems, we need a principled way of representing uncertainty quantitatively and manipulating and updating this representation in the light of observed data. Observed data is corrupted by noise and subject to measurement error, and many aspects of the phenomenon generating the data are usually imperfectly known if at all. The concept of uncertainty is also useful to justify the necessary abstraction of mapping a complex problem to a tractable domain of variables, accounting for our ignorance of the details.

Uncertainty is most conveniently represented by a probability distribution over the domain of variables of interest. A model induces a family of such global distributions via structural (conditional independence) constraints and specifications of local distribution families of simple well-understood form.[1] If a model is specified completely (i.e. could be used to generate data just like the true source), Bayes' formula can be used to compute marginal query probabilities conditioned on the observed data. We can distinguish between parametric and non-parametric models. In the former, the data generation mechanism is completely specified by a finite set of parameters and learning from data amounts to reducing our initial uncertainty about these, i.e. "fitting" the model to the observations by adjusting the parameters. In this thesis, we are mainly concerned with non-parametric methods whose setup is quite different. An important special case which serves

[1] Note that the domain often includes variables which cannot be observed (latent variables), because these lead to decompositions and therefore simplification of the model description.

well to illustrate the general idea is that of "smoothing" data by proposing an underlying curve which optimally solves a trade-off between fitting the data and minimising the "complexity" (quantified for example in terms of average bending energy). Non-parametric models can be constructed using random fields which (in contrast to the parametric situation) are not typically determined by a finite number of parameters. Nevertheless, if we are only interested in a finite-dimensional projection of such a field, inference becomes a finite problem and is either analytically tractable or amenable to common approximation techniques. Typically, this dimensionality is of the order of the training set size, in marked contrast to the parametric situation where the parameter dimensionality is usually independent of the training data.

This thesis contributes in two quite different aspects to the field of non-parametric modelling (more accurately to Gaussian process models introduced in Section 2.1). These are briefly motivated in the following subsections.

PAC-Bayesian Bounds for Gaussian Process Methods (Chapter 3)

In Chapter 3, we present a generalised and slightly improved version of McAllester's PAC Bayesian distribution-free generalisation error bound and provide a much simplified proof pointing out the basic concept clearly. We apply this result to Gaussian process classification methods.

Consider the binary classification (or pattern recognition) problem where data $S = \{(\boldsymbol{x}_i, y_i) \,|\, i = 1, \ldots, n\}$ is sampled *independent and identically distributed (i.i.d.)* from an unknown joint data distribution over input points $\boldsymbol{x} \in \mathcal{X}$ and targets $y \in \{-1, +1\}$. A good classification rule $r_S : \mathcal{X} \to \{-1, +1\}$ aims to minimise the *generalisation error* $\mathrm{gen}(S) = \Pr\{r_S(\boldsymbol{x}_*) \neq y_*\}$ if $(\boldsymbol{x}_*, y_*)$ is sampled from the data distribution independently of S. In statistical learning theory, the aim is to find distribution-free upper bounds $\mathrm{bound}(S, \delta)$ on $\mathrm{gen}(S)$ which are computable (i.e. independent of the data distribution which is unknown), such that the probability (over random draws of S) of a bound violation is $\leq \delta$. The important point about such a statement is that it holds no matter what the data

distribution is. Typically, $\text{bound}(S, \delta)$ converges to the *empirical error*

$$\text{emp}(S) = \frac{1}{n} \sum_{i=1}^{n} \text{I}_{\{r_S(\boldsymbol{x}_i) \neq y_i\}}$$

as $n \to \infty$, so that the bound statement implies convergence in probability of $\text{emp}(S)$ to $\text{gen}(S)$ and a bound on the rate of convergence. If this is fast enough, one obtains almost sure convergence, thus the consistency of the method. Comparing rate bounds for different methods or for the same method under different configurations can help to make design choices or even drive model selection. In critical high-security applications, the existence of a useful distribution-free bound can be reassuring. In this thesis, we present generalisations and simplifications of a powerful general data-dependent technique to prove distribution-free upper bounds for Bayesian-type classification rules, and we apply this technique to non-parametric Bayesian Gaussian process classification methods. The main concept we require is an inequality coming from a notion of duality in convex analysis. Compared with techniques which have been used in the past to prove similar distribution-free bounds, our proof methods are surprisingly light-weight and efficient. As opposed to many "uniform" results, our bounds strongly depend on prior knowledge encoded into the procedure: predictors learned from samples S which are very unlikely under the prior assumptions are judged more complex and suffer a larger penalty in the bound than predictors from samples conforming to our assumptions.

Sparse Gaussian Process Methods (Chapter 4)

In Chapter 4, we present sparse approximation schemes for Gaussian process methods which often show performance similar to standard GP inference approximations, but have vastly reduced time and memory requirements.

Recall that although practical non-parametric methods have finite-dimensional uncertainty representations, these usually scale with the training set size $n = |S|$, leading to memory and training time scaling superlinear (typically at least quadratic) in n and prediction time scaling of at least $O(n)$. Yet if there is to be any hope of useful generalisation from a source, data must exhibit redundancies as

n grows large, and in the second part of this thesis we describe a general way of detecting such redundancies and using them in "sparse" variants of non-parametric methods to achieve a much more favourable scaling of time and memory requirements. The concepts we use are a generic framework for approximate inference in Gaussian process (GP) models combined with greedy forward selection of "informative" cases from S via information-theoretic criteria from "active learning" (sequential design). Our contributions here might appeal more to the practitioner than the theoretician. We present a clear and simple discussion of the concepts underlying sparse approximations. The concrete algorithms are modifications of a non-sparse generic approximate inference technique which are motivated by the analysis. They span the most interesting cases arising in practice, and the primitives that can be specialised to obtain new variants are pointed out clearly. We place special emphasis on numerically stable algorithms with a favourable tradeoff between memory and running time requirements whose dominating operations can be vectorised to make use of highly efficient numerical linear algebra software. Most importantly, our framework is complete in that it includes automatic empirical Bayesian model selection. Its tractability and efficiency is demonstrated on a significant number of large tasks.

Background (Chapter 2)

Apart from scientifically novel contributions, we have included a range of tutorial material which we hope the reader will find a further useful contribution. The thesis is entirely self-contained along its main branches. Chapter 2 contains background material, notably an extensive discussion (Section 2.1) of notions of Gaussian processes relevant for many machine learning techniques. Section 2.2 gives a brief description of some key concepts of learning theory where we have tried to emphasise intuitive ideas rather than technical details.

Software

A large amount of C++ software has been developed in order to run the experiments presented in this thesis (among others). Strictly speaking, this is not a complete contribution at this time since the package has not been released into the public domain yet.[2] The initial reason to implement this system was my frustration with the widely used Matlab prototyping system. The main drawbacks of Matlab, apart from its high pricing[3], are its poor memory management and its incredible slowness for iterative code. The latter can be circumvented by implementing C++ plug-ins, but for the former it seems impossible to do any sort of non-trivial operations on matrices without producing two or three copies. Also, in order to circumvent slow iterative code, artificial matrices of simple (low-rank) structure have to be created temporarily to be able to use vectorised operations. Finally, although Matlab contains a host of powerful functions, none of them are easily extensible. The system does not allow for proper object-oriented development which makes life so much easier w.r.t. extensibility.

Our system contains an extensive matrix and vector library which allows simple manipulations of parts or permutations of matrices *without* the need to make copies. A module for generating pseudo-random variates from all commonly used distributions is available. Several powerful gradient-based nonlinear optimisers are included which allow direct control of the optimisation process (non-standard scenarios such as the one required in Section 4.5.3 are easily accommodated). A module for dataset import and export from different formats is designed to allow for virtual "views" on large datasets which are accessed via intermediate memory buffers,[4] and virtual datasets can be represented as indices into these views. Generic implementations of frequently used experimental setups (for example,

[2]This will hopefully happen in the near future, if I can figure out a good way of marrying the code with Matlab. Even as stand-alone, many small things have to be tidied up. Some of the code unfortunately uses the *Numerical Recipes* distribution [145] which is proprietary. Another important point is the re-design of the matrix library to conform to the Fortran storage model, so that optimised numerical software can be linked in.

[3]For example, essential basic features are packaged in "toolboxes" which have to be bought separately. The basic installation cannot even be used to sample Gamma variables or do basic data analysis.

[4]These features are not fully implemented.

model selection by cross-validation) can work with arbitrary "learning methods", produce "prediction value vectors" of arbitrary format which can then be evaluated against a test dataset using arbitrary "loss functions". We use our own data file formats for which Matlab functions for read and write access are available, so results can be visualised in Matlab. We have implemented a rudimentary interface which allows our system to be controlled interactively from Matlab. This interface is object-oriented and works by file-based transfer and file semaphores, running our system as a server process.

1.1 Declaration of Previous Work, Collaborations

Parts of the material presented in this thesis has been done in collaboration with other researchers and has been previously published (a list of publications related to this thesis is given below).

Most material in Chapter 3 appeared previously in [170] and the accompanying report [171]. I became interested in the PAC-Bayesian theorem through work with John Langford and Nimrod Megiddo [172]. I am grateful to Manfred Opper for inviting me to a seminar at Aston University, Birmingham. The discussions I had with Manfred led to great simplifications in the argumentation, and the accessibility of this chapter owes a lot to these. I would also like to thank John Shawe-Taylor for organising the NeuroCOLT workshop on "bounds below 1/2" in Windsor and inviting me, and to David McAllester, John Langford, Manfred Opper and others for inspiring discussions at the "Royal" venue. Also thanks to Ralf Herbrich for many discussions about learning theory and other issues during my time at MS Research, Cambridge.

Material in Chapter 4 appeared previously in [98] and [165], although a significant part (especially about model selection and PLV) is novel. The published work has been done in collaboration with Neil Lawrence and Ralf Herbrich for the IVM[5] (based on the presentation [97]) and with Christopher Williams and Neil Lawrence for the PLV regression scheme. Chris Williams sparked my interest for

[5]In particular, I do not bear any responsibility for the name "informative vector machine" (or would have chosen a longer and more boring one).

sparse approximations with our work [211, 208] on Nyström approximations. As discussed in detail in the chapter, both our schemes are closely related to a body of work done by Lehel Csató and Manfred Opper.

1.1.1 Publications during Postgraduate Studies

In this section, I list work published during my postgraduate studies (in chronological order). Some of the work is not described in this thesis, and in these cases I have added short abstracts here.

- Matthias Seeger, Christopher Williams, Neil Lawrence
  Fast Forward Selection to Speed Up Sparse Gaussian Process Regression
  Workshop on AI and Statistics 9 (2003). [165]

- Neil Lawrence, Matthias Seeger, Ralf Herbrich
  Fast Sparse Gaussian Process Methods: The Informative Vector Machine
  Neural Information Processing Systems 15 (2003). [98]

- Matthias Seeger
  PAC-Bayesian Generalization Error Bounds for Gaussian Process Classification
  Journal of Machine Learning Research 3 (2002), 233–269. [170]

- Matthias Seeger
  Covariance Kernels from Bayesian Generative Models
  Neural Information Processing Systems 14 (2002), 905–912. [164]
  We propose *mutual information kernels* as general framework of learning covariance functions for GP models from labelled and unlabelled task data (related to the Fisher kernel and Haussler's HMRF kernels discussed in Section 2.1.8). We show how such kernels can be evaluated approximately by variational techniques and present experiments with variational Bayesian mixtures of factor analysers.

- Matthias Seeger, John Langford, Nimrod Megiddo
  An Improved Predictive Accuracy Bound for Averaging Classifiers

International Conference on Machine Learning 18 (2001), 290–297. [172]
Combines McAllester's PAC Bayesian theorem with the approximation technique for mixture discriminants suggested in [158] to obtain a PAC Bayesian bound for Bayes-type classifiers.

- Christopher Williams, Matthias Seeger
  Using the Nyström Method to Speed Up Kernel Machines
  Neural Information Processing Systems 13 (2001), 682–688. [211]

- Christopher Williams, Matthias Seeger
  The Effect of the Input Density Distribution on Kernel-based Classifiers
  International Conference on Machine Learning 17 (2000), 1159–1166. [208]

- Matthias Seeger
  Learning with Labelled and Unlabelled Data
  Technical report, University of Edinburgh (2001).
  See `www.dai.ed.ac.uk/∼seeger/`.
  Provides a rigorous formal definition of the problem of learning from a mixture of labelled and unlabelled data, a literature review fairly complete at that time and the development of the framework of input-dependent regularisation.

- Matthias Seeger
  Input-dependent Regularization of Conditional Density Models
  Technical report, University of Edinburgh (2001).
  See `www.dai.ed.ac.uk/∼seeger/`.
  Proposes a general statistical framework for learning from labelled and unlabelled data together with some ideas for algorithms based on standard latent variable techniques similar to EM.

- Matthias Seeger
  Annealed Expectation-Maximization by Entropy Projection
  Technical report, University of Edinburgh (2000).
  See `www.dai.ed.ac.uk/∼seeger/`.

Complements the frequently used technique of deterministic annealing by *M-step annealing* which can be used to learn assignment structure in general "resources-components" models. An application to Gaussian mixture modelling with tied covariance matrix parameters is given.

Chapter 2

Background

The statistical models we are concerned with in this thesis are non-parametric. By employing Gaussian processes (GPs) as prior distributions over latent fields, these models and inference techniques for them can be understood from a conceptually simple Bayesian viewpoint. In Section 2.1 we give an introduction to GPs and their role in non-parametric regression models. Chapter 3 of this thesis is concerned with learning-theoretical finite sample size analyses of Bayesian GP methods, and concepts and methodology required there is introduced in Section 2.2. We hope that both sections will be useful for readers not familiar with these subjects to grasp the basic concepts. Details are largely omitted, but references are provided.

2.1 Bayesian Gaussian Processes

Gaussian processes (GPs) are natural generalisations of multivariate Gaussian random variables to infinite (countably or continuous) index sets. GPs have been applied in a large number of fields to a diverse range of ends, and very many deep theoretical analyses of various properties are available. On the other hand, in this thesis we only require quite elementary properties and none of the deep theory behind GPs. Although our exposition in this section remains selective and on a fairly elementary and non-rigorous level, it contains much material beyond what is required to follow the remainder of this thesis. Readers from the machine learning

community might find our introduction valuable since it tries to connect different branches in which similar ideas have been investigated and phrases concepts in the familiar vocabulary of statistical modelling. On the other hand, the section can safely be skipped on the first reading and visited later for references.

Other introductions to GP models in the machine learning context are given in [111, 132, 209, 146].

2.1.1 Gaussian Processes: The Process and the Weight Space View

Gaussian process (GP) models are constructed from classical statistical models by replacing latent functions of parametric form by random processes with Gaussian prior. In this section, we will introduce GPs and highlight some aspects which are relevant to this thesis. We develop two simple views on GPs, pointing out similarities and key differences to distributions induced by parametric models. We follow [2], Chap. 1,2. A good introduction into the concepts required to study GP prediction is given in [189], Chap. 2. For concepts and vocabulary from general probability theory, we refer to [67, 16, 29]. The section can be skipped by readers not interested in details or familiar with GPs, maybe with a quick glance of what we mean by process and weight space view.

Let $\{X_n\}$ be a sequence of complex-valued random variables, and recall that $X_n \to X$ $(n \to \infty)$ *in quadratic mean* (or *in mean square (m.s.)*) if $\mathrm{E}[|X_n - X|^2] \to 0$. M.s. convergence is weaker than *almost sure (a.s.)* convergence, but turns out to be the most useful mode for discussing GP aspects we require here. In general, X and Y are *m.s. equivalent* if $\mathrm{E}[|X - Y|^2] = 0$. Let $\mathcal{X}$ be an non-empty index set. For the main parts of this thesis, $\mathcal{X}$ can be arbitrary, but here we assume that $\mathcal{X}$ is at least a group[1] (and sometimes we assume it to be $\mathbb{R}^d$). In a nutshell, a *random process* $\mathcal{X} \to \mathbb{C}$ is a collection of random variables (one for each $\boldsymbol{x} \in \mathcal{X}$) over a common probability space. The measure-theoretic definition is awkward, but basically the same as for a single variable. A process can be defined via its *finite-dimensional distributions (f.d.d.'s)* which it induces on each

[1] Has an addition $+$, an origin $\mathbf{0}$ and a negation $-$.

finite subset of $\mathcal{X}$. Kolmogorov [90] proved that for a specification of all f.d.d.'s there exists a random process (in the measure-theoretic sense) iff the f.d.d.'s are *symmetric* and *consistent*. Namely, for every $n \in \mathbb{N}_{>0}$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, Borel sets $B_1, \ldots, B_n$ and every permutation π of $\{1, \ldots, n\}$ we must have

$$\mu_{\boldsymbol{x}_{\pi(1)},\ldots,\boldsymbol{x}_{\pi(n)}}(B_{\pi(1)} \times \cdots \times B_{\pi(n)}) = \mu_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n}(B_1 \times \cdots \times B_n) \quad \text{and}$$
$$\mu_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n}(B_1 \times \cdots \times B_{n-1} \times \mathbb{C}) = \mu_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{n-1}}(B_1 \times \cdots \times B_{n-1}).$$

The question about uniqueness of random processes is tricky, because two processes can be *equivalent* ($u(\boldsymbol{x}) = v(\boldsymbol{x})$ almost surely for every $\boldsymbol{x}$; equivalent processes are called *versions* of each other), yet differ significantly w.r.t. almost sure properties of their sample paths. We do not require the study of sample path properties in this thesis, but see [2] for more information.

The first and second-order statistics of $u(\boldsymbol{x})$ are its *mean function* $m(\boldsymbol{x}) = \mathrm{E}[u(\boldsymbol{x})]$ and *covariance function*[2]

$$K(\boldsymbol{x}, \boldsymbol{x}') = \mathrm{E}\left[(u(\boldsymbol{x}) - m(\boldsymbol{x}))\overline{(u(\boldsymbol{x}) - m(\boldsymbol{x}))}\right].$$

The latter is central to study characteristics of the process in the mean-square sense. It is a *positive semidefinite*[3] form in the sense that for every $n \in \mathbb{N}$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathcal{X}$, $z_1, \ldots, z_n$:

$$\sum_{i,j=1}^{n} z_i \overline{z_j} K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0. \tag{2.1}$$

This is clear because for $X = \sum_i z_i(u(\boldsymbol{x}_i) - m(\boldsymbol{x}_i))$ we have $\mathrm{E}[|X|^2] \geq 0$. Note that this implies that $K(\boldsymbol{x}, \boldsymbol{x}) \geq 0$ and $K(\boldsymbol{x}', \boldsymbol{x}) = \overline{K(\boldsymbol{x}, \boldsymbol{x}')}$. K is called *positive definite* if (2.1) holds with $>$ whenever $\boldsymbol{z} \neq \boldsymbol{0}$. The positive semidefiniteness of K leads to an important spectral decomposition which is discussed in Section 2.1.4. A positive semidefinite K will also be referred to as *kernel*, pointing out its role as kernel for a linear operator (see Section 2.1.4).

[2]Here, $\overline{a + ib} = a - ib$, $a, b \in \mathbb{R}$ denotes the complex conjugate.

[3]This term is not uniquely used in the literature, it is sometimes replaced by *non-negative definite* or even *positive definite* (which has a different meaning here). A different weaker property is *conditional positive semidefiniteness* (see [200], Chap. 2) which is important but not discussed here.

2.1.1.1 Stationary Processes

In many situations, the behaviour of the process does not depend on the location of the observer, and under this restriction a rich theory can be developed, linking local m.s. properties of the process to the behaviour of K close to the origin. A process is called *strictly homogeneous* (or *strictly stationary*) if its f.d.d.'s are invariant under simultaneous translation of their variables. This implies that $m(\boldsymbol{x})$ is constant and $K(\boldsymbol{x}, \boldsymbol{x}')$ is a function of $\boldsymbol{x} - \boldsymbol{x}'$ (we write $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x} - \boldsymbol{x}')$ in this case; $K(\boldsymbol{x})$ is sometimes called *autocovariance function*). A process fulfilling the latter two conditions is called *(weakly) homogeneous* (or *(weakly) stationary*). For a stationary process, the choice of the origin is not reflected in the statistics up to second order. If $K(\mathbf{0}) > 0$,

$$\rho(\boldsymbol{x}) = \frac{K(\boldsymbol{x})}{K(\mathbf{0})}$$

is called *correlation function*. A stationary process has a spectral representation as a stochastic Fourier integral (e.g., [2], Chap. 2; [67], Chap. 9; [214]), based on *Bochner's theorem* which (for $\mathcal{X} = \mathbb{R}^d$) asserts that $\rho(\boldsymbol{x})$ is positive semidefinite, furthermore uniformly continuous with $\rho(\mathbf{0}) = 1$, $|\rho(\boldsymbol{x})| \leq 1$ iff it is the characteristic function of a variable $\boldsymbol{\omega}$, *i.e.*

$$\rho(\boldsymbol{x}) = \int e^{i\,\boldsymbol{x}^T\boldsymbol{\omega}}\, dF(\boldsymbol{\omega}) \tag{2.2}$$

for a probability distribution function $F(\boldsymbol{\omega})$. If $F(\boldsymbol{\omega})$ has a density $f(\boldsymbol{\omega})$ (w.r.t. Lebesgue measure), f is called *spectral density*.[4] This theorem allows to prove positive semidefiniteness of K by computing its Fourier transform and checking that it is non-negative. If so, it must be the spectral density. Note that if $\rho(\boldsymbol{x})$ is real (thus an odd function), then the spectral distribution is symmetric around $\mathbf{0}$, and if $f(\boldsymbol{\omega})$ exists it is odd as well.

The f.d.d.'s of a process determine its mean-square properties, while this is not true in general for almost sure properties. Even stronger, for a zero-mean process, m.s. properties are usually determined entirely by the covariance function

[4] If $\rho(\mathbf{0}) > 0$, the spectral distribution is a measure of total mass $\rho(\mathbf{0})$.

$K(\boldsymbol{x}, \boldsymbol{x}')$. For stationary processes, it is merely the behaviour of $K(\boldsymbol{x})$ at the origin which counts: the m.s. derivative[5] $D_{\boldsymbol{x}} u(\boldsymbol{x})$ exists everywhere iff $D_{\boldsymbol{x}} D_{\boldsymbol{x}} K(\boldsymbol{x})$ exists at $\boldsymbol{x} = \mathbf{0}$. For example, a process with the RBF (Gaussian) covariance function K (2.29) is m.s. analytic, because K is analytic (differentiable up to any order) at $\mathbf{0}$.

2.1.1.2 Isotropic Processes

A stationary process is called *isotropic* if its covariance function $K(\boldsymbol{x})$ depends on $\|\boldsymbol{x}\|$ only. In this case, the spectral distribution F is invariant under isotropic isomorphisms (e.g., rotations). Loosely speaking, second-order characteristics of an isotropic process are the same from whatever position and direction they are observed. It is much simpler to characterise isotropic correlation functions than stationary ones in general. Let $\rho(\tau) = \rho(\boldsymbol{x})$ for $\tau = \|\boldsymbol{x}\|$. The spectral decomposition (2.2) simplifies to

$$\rho(\tau) = \int \Lambda_{d/2-1}(\tau\,\omega) dF(\omega) \tag{2.3}$$

where $F(\omega) = \int \mathrm{I}_{\{\|\boldsymbol{\omega}\| \leq \omega\}}\, dF(\boldsymbol{\omega})$ is a distribution function and Λ_ν is an expression involving a Bessel function of the first kind (see [2], Sect. 2.5). Alternatively, if the spectral density $f(\boldsymbol{\omega})$ exists and $f(\omega) = f(\boldsymbol{\omega})$ for $\omega = \|\boldsymbol{\omega}\|$, then $dF(\omega) = A_{d-1}\omega^{d-1} f(\omega)\, d\omega$,[6] so we can easily convert to the spectral representation in terms of $f(\omega)$. Denote the set of $\rho(\tau)$ corresponding to isotropic correlation functions in $\mathbb{R}^d$ by $\mathcal{D}^d$. Note that (2.3) characterises $\mathcal{D}^d$ (by Bochner's theorem). Also, it is clear that $\mathcal{D}^{d+1} \subset \mathcal{D}^d$.[7] Let $\mathcal{D}^\infty = \bigcap_{d \geq 1} \mathcal{D}^d$. Since

$$\Lambda_{d/2-1}(x) \to e^{-x^2},\ d \to \infty,$$

one can show that $\rho(\tau) \in \mathcal{D}^\infty$ iff $\rho(\tau) = \int \exp(-\tau^2\omega^2)\, dF(\omega)$. Note that the assumption of isotropy puts strong constraints on the correlation function, especially for large d. For example, $\rho(\tau) \geq \inf_x \Lambda_{d/2-1}(x) \geq -1/n$ so large negative

[5] Here, $D_{\boldsymbol{x}}$ denotes a differential functional, such as $\partial^2/(\partial x_1 \partial x_2)$.

[6] A_{d-1} is the surface area of the unit sphere in $\mathbb{R}^d$.

[7] Beware that both $F(\omega)$ and $f(\omega)$ depend on the dimension d for which $\rho(\tau)$ is used to induce a correlation function.

correlations are ruled out. If $\rho(\tau) \in \mathcal{D}^\infty$, it must be non-negative. Furthermore, for large d $\rho(\tau)$ is smooth on $(0, \infty)$ while it may have a jump at 0 (additive white noise). Also, $\rho(\tau)$ converges for $\tau \to \infty$ so we can restrict our attention to $\rho(\tau)$ which vanish for large τ. If $\rho(\tau) \in \mathcal{D}^d$ and $\boldsymbol{B} \in \mathbb{R}^{d,d}$ is nonsingular, then

$$\rho_{\boldsymbol{B}}(\boldsymbol{x}) = \rho(\|\boldsymbol{B}\boldsymbol{x}\|)$$

is a correlation function as well, called *anisotropic*. Examples of (an)isotropic covariance functions are given in Section 2.1.8.

2.1.1.3 Two Views on Gaussian Processes

A *Gaussian process (GP)* is a process whose f.d.d.'s are Gaussian. Since a Gaussian is determined by its first and second-order cumulants and these involve pairwise interactions only, its f.d.d.'s are completely determined by mean and covariance function. GPs are by far the most accessible and well-understood processes (on uncountable index sets). It is clear that for every positive semidefinite function K there exists a zero-mean GP with K as covariance function, so GPs as modelling tool are very flexible. In conjunction with latent variable modelling techniques, a wide variety of non-parametric models can be constructed (see Section 2.1.2). The fact that all f.d.d.'s are Gaussian with covariance matrices induced by $K(\boldsymbol{x}, \boldsymbol{x}')$ can be used to obtain approximations to Bayesian inference fairly straightforwardly (see Section 2.1.3). It is interesting to note that derivatives $D_{\boldsymbol{x}} u(\boldsymbol{x})$ of a GP are GPs again (if they exist), and

$$\mathrm{E}\left[D^{(1)}_{\boldsymbol{x}} u(\boldsymbol{x}) \overline{D^{(2)}_{\boldsymbol{x}'} u(\boldsymbol{x}')}\right] = D^{(1)}_{\boldsymbol{x}} D^{(2)}_{\boldsymbol{x}'} K(\boldsymbol{x}, \boldsymbol{x}'),$$

thus derivative observations can be incorporated into a model in the same way as function value observations (for applications, see [136, 184]). Characteristics such as m.s. differentiability up to a given order can be controlled via the covariance function, an example is given in Section 2.1.8. For example, one of the most thoroughly studied GPs is the *Wiener process* (or *Brownian motion*, or *continuous random walk*) with covariance function $K(x, x') = \sigma^2 \min\{x, x'\}$ (here, $\mathcal{X} = \mathbb{R}_{\geq 0}$; for multivariate generalisations to Brownian sheets, see [2], Chap. 8).

$u(x)$ is m.s. continuous everywhere, but not m.s. differentiable anywhere. In fact, a version of the Wiener process can be constructed which has continuous sample paths, but for every version sample paths are nowhere differentiable with probability 1. The Wiener process can be used to construct other GPs by means of stochastic integrals (e.g., [67], Chap. 13).

We now develop two elementary views on Gaussian processes, the *process* and the *weight space view*. While the former is usually much simpler to work with (and will be used almost exclusively in this thesis), the latter allows us to relate GP models with parametric linear models rather directly. We follow [209].[8] The process view on a zero-mean GP $u(\boldsymbol{x})$ with covariance function K is in the spirit of the GP definition given above. $u(\boldsymbol{x})$ is defined implicitly, in that for any finite subset $X \subset \mathcal{X}$ it induces a f.d.d. $N(\boldsymbol{0}, \boldsymbol{K}(X))$ over the vector $\boldsymbol{u} = u(X)$ of process values at the points X. Here, $\boldsymbol{K}(X) = \boldsymbol{K}(X, X) = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j}$ where $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. Kolmogorov's theorem guarantees the existence of a GP with this family of f.d.d.'s.[9] In practice, many modelling problems involving an unknown functional relationship $u(\boldsymbol{x})$ can be formulated such that only ever a finite number of linear characteristics of $u(\boldsymbol{x})$ (e.g., evaluations or derivatives of $u(\boldsymbol{x})$) are linked to observations or predictive queries, and in such cases the process view boils down to dealing with the "projection" of the GP onto a multivariate Gaussian distribution, thus to simple linear algebra of quadratic forms.

GPs can also be seen from a *weight space viewpoint*, relating them to the linear model. In the Bayesian context this view was first suggested by O'Hagan [135] as a "localized regression model" (the weight space is finite-dimensional there) while the generalisation to arbitrary GP priors developed there uses the process view. This paper is among the first to address GP regression in a rigorous Bayesian context, while the equivalence between spline smoothing and Bayesian estimation of processes was noticed earlier by Kimeldorf and Wahba [89] (see Section 2.1.5).

[8] We use the term "process view" instead of "function space view" employed in [209]. The relationship between GPs and associated spaces of smooth functions is a bit subtle and introduced only below in Section 2.1.4.

[9] If K is continuous everywhere, a version exists with continuous sample paths, but we do not require this here.

Recall the linear model

$$y = \Phi(\boldsymbol{x})^T \boldsymbol{\beta} + \varepsilon, \tag{2.4}$$

where $\Phi(\boldsymbol{x})$ is a feature map from the covariate $\boldsymbol{x}$ and ε is independent Gaussian noise. Every GP whose covariance function satisfies weak constraints can be written as (2.4), albeit with possibly infinite-dimensional weight space. To develop this view, we use some details which are discussed in detail below in Section 2.1.4. Under mild conditions on the covariance function $K(\boldsymbol{x}, \boldsymbol{x}')$ of $u(\boldsymbol{x})$, we can construct a sequence

$$\sum_{\nu=1}^{k} \beta_\nu \lambda_\nu^{1/2} \phi_\nu(\boldsymbol{x}),$$

which converges to $u(\boldsymbol{x})$ in quadratic mean ($k \to \infty$).[10] Here, β_ν are i.i.d. $N(0,1)$ variables. ϕ_ν are orthonormal eigenfunctions of an operator induced by K with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$, $\sum_{\nu \geq 1} \lambda_\nu^2 < \infty$, in a sense made precise in Section 2.1.4. Thus, if $\boldsymbol{\beta} = (\beta_\nu)_\nu$ and $\Phi(\boldsymbol{x}) = (\lambda_\nu^{1/2} \phi_\nu(\boldsymbol{x}))_\nu$, then $u(\boldsymbol{x}) = \Phi(\boldsymbol{x})^T \boldsymbol{\beta}$ in quadratic mean, and $\Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$. This is the *weight space view* on GPs and allows to view a non-parametric regression model

$$y = u(\boldsymbol{x}) + \varepsilon$$

as direct infinite-dimensional generalisation of the linear model (2.4) with spherical Gaussian prior on $\boldsymbol{\beta}$. We say that $\Phi(\boldsymbol{x})$ maps into a *feature space* which is typically (countably) infinite-dimensional. It is important to note that in this construction of the feature map $\Phi(\boldsymbol{x})$ the individual components $\lambda_\nu^{1/2} \phi_\nu(\boldsymbol{x})$ do not have the same scaling, in the sense that their norm in $\mathcal{L}_2(\mu)$ (the Hilbert space they are drawn from and that K operates on) is $\lambda_\nu^{1/2} \to 0$ ($\nu \to \infty$). They are comparable in a different (RKHS) norm which scales with the "roughness" of a function. Intuitively, as $\nu \to \infty$, the graph of ϕ_ν becomes rougher and increasingly complicated, see Section 2.1.4 for details.

For all inference purposes which are concerned with f.d.d.'s of $u(\boldsymbol{x})$ and its derivatives (or other linear functionals) only, the process and the weight space

[10] We only need pointwise m.s. convergence, although much stronger statements are possible under mild assumptions, e.g. [2], Sect. 3.3.

view are equivalent: they lead to identical results. However, we feel that often the process view is much simpler to work with, avoiding spurious infinities[11] and relying on familiar Gaussian manipulations only. On the other hand, the weight space view is more frequently used at least in the machine learning literature, and its peculiarities may be a reason behind the perception that GP models are difficult to interpret. There is also the danger that false intuitions or conclusions are developed from interpolating geometrical arguments from low-dimensional Euclidean space to the feature space. We should also note that a weight space representation of a GP in terms of a feature map Φ is not unique. The route via eigenfunctions of the covariance operator is only one way to establish such.[12] About the only invariant is that we always have $\Phi(\boldsymbol{x})^T\Phi(\boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}')$.

2.1.1.4 Gaussian Processes as Limit Priors of Parametric Models

We conclude this section by mentioning that one of the prime reasons for focusing current machine learning interest on GP models was a highly original different way of establishing a weight space view proposed in [131]. Consider a model

$$f(\boldsymbol{x}) = \sum_{j=1}^{H} v_j h(\boldsymbol{x}; \boldsymbol{u}^{(j)})$$

which could be a multi-layer perceptron (MLP) with hidden layer functions h, weights $\boldsymbol{u}^{(j)}$ and output layer weights $\boldsymbol{v}$. Suppose that $\boldsymbol{u}^{(j)}$ have independent identical priors s.t. the resulting $h(\boldsymbol{x}; \boldsymbol{u}^{(j)})$ are bounded almost surely over a compact region of interest. Also, $v_j \sim N(0, \omega^2/H)$ independently. Then, for $H \to \infty$, $f(\boldsymbol{x})$ converges in quadratic mean to a zero-mean GP with covariance function $\omega^2 \mathrm{E}_{\boldsymbol{u}}[h(\boldsymbol{x}; \boldsymbol{u})h(\boldsymbol{x}'; \boldsymbol{u})]$. Stronger conditions would assure almost sure convergence uniformly over a compact region. The bottom line is that if we take a conventional parametric model which linearly combines the outputs of a large number of feature detectors, and if we scale the outputs s.t. each of them in

[11] Which seem to drop out almost "magically" in the end from the weight space viewpoint, while infinities do not occur in the process view.

[12] For example, in Section 2.1.4 we discuss K's role as reproducing kernel, in the sense that $K(\boldsymbol{x}, \boldsymbol{x}') = (K(\cdot, \boldsymbol{x}), K(\cdot, \boldsymbol{x}'))_K$ in some Hilbert space with inner product $(\cdot, \cdot)_K$. We could define Φ to map $\boldsymbol{x} \mapsto K(\cdot, \boldsymbol{x})$ and use the Hilbert space as weight space.

isolation has only a negligible contribution to the response, we might just as well use the corresponding Gaussian process model. Neal [131] also shows that if a non-zero number of the non-Gaussian feature outputs have a significant impact on the response with non-zero probability, then the limit process is typically not Gaussian.

To conclude, the weight space view seems to relate non-parametric GP models with parametric linear models fairly directly. However, there are important differences in general. Neal showed that GPs are obtained as limit distributions of large linear combinations of features if each feature's contribution becomes negligible, while the output distributions of architectures which fit at least a few strong feature detectors are typically not Gaussian. Predictions from a GP model are smoothed versions of the data (in a sense made concrete in Section 2.1.5), *i.e.* interpolate by minimising general smoothness constraints encoded in the GP prior, as opposed to parametric models which predict by focusing on these functions (within the family) which are most consistent with the data. O'Hagan [135] discusses differences w.r.t. optimal design.

2.1.2 Some Gaussian Process Models

The simplest Gaussian process model is

$$y = u + \varepsilon,$$

where $u = u(\boldsymbol{x})$ is *a priori* a zero-mean Gaussian process with covariance function K and ε is independent $N(0, \sigma^2)$ noise. Inference for this model is simple and analytically tractable, because the observation process $y(\boldsymbol{x})$ is zero-mean Gaussian with covariance $K(\boldsymbol{x}, \boldsymbol{x}') + \sigma^2 \delta_{\boldsymbol{x},\boldsymbol{x}'}$.[13] Given some i.i.d. data $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\}$, let $\boldsymbol{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j}$. Then, $P(\boldsymbol{u}) = N(\boldsymbol{0}, \boldsymbol{K})$ and

$$P(\boldsymbol{u}|S) = N\left(\boldsymbol{K}(\sigma^2 \boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{y}, \sigma^2(\sigma^2 \boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{K}\right), \tag{2.5}$$

[13] In the context of this model, it is interesting to note that if K' is stationary and continuous everywhere except at $\boldsymbol{0}$, it is the sum of a continuous (stationary) covariance K and a white noise covariance $\propto \delta_{\boldsymbol{x},\boldsymbol{x}'}$. Furthermore, Schönberg conjectured that if K' is an isotropic bounded covariance function, it must be continuous except possibly at $\boldsymbol{0}$.

where $\boldsymbol{u} = (u(\boldsymbol{x}_i))_i$. For some test point $\boldsymbol{x}_*$ distinct from the training points, $u_* = u(\boldsymbol{x}_*) \perp \boldsymbol{y} \mid \boldsymbol{u}$, so that

$$P(u_*|\boldsymbol{x}_*, S) = \int P(u_*|\boldsymbol{u})P(\boldsymbol{u}|S)\, d\boldsymbol{u}$$
$$= N\left(u_*|\boldsymbol{k}(\boldsymbol{x}_*)^T(\sigma^2\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{y}, K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}(\boldsymbol{x}_*)^T(\sigma^2\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{k}(\boldsymbol{x}_*)\right).$$

Here, $\boldsymbol{k}(\boldsymbol{x}_*) = (K(\boldsymbol{x}_i, \boldsymbol{x}_*))_i$, and we used the tower properties (Lemma A.3) and Gaussian formulae (Section A.4.3). We see that for this model, the *posterior predictive process* $u(\boldsymbol{x})$ given S is Gaussian with mean function $\boldsymbol{y}^T(\sigma^2\boldsymbol{I}+\boldsymbol{K})^{-1}\boldsymbol{k}(\boldsymbol{x})$ and covariance function

$$K(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}(\boldsymbol{x})^T(\sigma^2\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{k}(\boldsymbol{x}').$$

In practice, if only posterior mean predictions are required, the prediction vector $\boldsymbol{\xi} = (\sigma^2\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{y}$ can be computed using a linear conjugate gradients solver which runs in $O(n^2)$ if the eigenvalue spectrum of $\boldsymbol{K}$ (which has to be stored explicitly) shows a fast decay. If predictive variances for many test points are required, the Cholesky decomposition (Definition A.2) $\sigma^2\boldsymbol{I} + \boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^T$ should be computed, after which each variance computation requires a single back-substitution.

The pointwise predictive variance is never larger than the corresponding prior variance, but the shrinkage decreases with increasing noise level σ^2. The same result can be derived in the weight space view with $u(\boldsymbol{x}) = \Phi(\boldsymbol{x})^T\boldsymbol{\beta}$, applying the standard derivation of Bayesian linear regression (e.g., [209]). Note that just as in parametric linear regression, the smoothed prediction $\mathrm{E}[\boldsymbol{y}|S]$ is a linear function of the observations $\boldsymbol{y}$, as is the mean function of the predictive process $\mathrm{E}[u(\boldsymbol{x})|S]$ (see also Section 2.1.7). Note also that if $K(\boldsymbol{x}, \boldsymbol{x}') \to 0$ as $\|\boldsymbol{x} - \boldsymbol{x}'\|$ gets big, predictive mean and variance for points $\boldsymbol{x}$ far from all data tend to prior mean 0 and prior variance $K(\boldsymbol{x}, \boldsymbol{x})$. Second-level inference problems such as selecting values for hyperparameters (parameters of K and σ^2) or integrating them out are not analytically tractable and approximations have to be applied. Approximate model selection is discussed in Section 2.1.3.

We can generalise this model by allowing for an arbitrary "noise distribution" $P(y|u)$, retaining the GP prior on $u(\boldsymbol{x})$. The generative view is to sample the

process $u(\cdot)$ from the prior, then $y_i \sim P(y_i|u(\boldsymbol{x}_i))$ independent from each other given $u(\cdot)$.[14] The likelihood function factors as a product of univariate terms:

$$P(\boldsymbol{y}|\boldsymbol{X}, u(\cdot)) = P(\boldsymbol{y}|\boldsymbol{u}) = \prod_{i=1}^{n} P(y_i|u_i). \tag{2.6}$$

Since the likelihood depends on $u(\cdot)$ only via the finite set $\boldsymbol{u}$, the predictive posterior process can be written as

$$dP(u(\cdot)|S) = \frac{P(\boldsymbol{u}|S)}{P(\boldsymbol{u})} dP(u(\cdot)), \tag{2.7}$$

thus the prior measure is "shifted" by multiplication with $P(\boldsymbol{u}|S)/P(\boldsymbol{u})$ depending on the process values $\boldsymbol{u}$ only (recall our notation from Section A.1). The predictive process is not Gaussian in general, but its mean and covariance function can be obtained from knowledge of the posterior mean and covariance matrix of $P(\boldsymbol{u}|S)$ as discussed in Section 2.1.3. For a test point $\boldsymbol{x}_*$,

$$P(y_*|\boldsymbol{x}_*, S) = \mathrm{E}\left[P(y_*|u_*)\right]$$

where the expectation is over the predictive distribution of u_*. In this general model, first-level inference is not analytically tractable. In Section 2.1.3 a general approximate inference framework is discussed. MCMC methods (see Section A.6.3) can be applied fairly straightforwardly, for example by Gibbs sampling from the latent variables $\boldsymbol{u}$ [132]. Such methods are attractive because the marginalisation over hyperparameters can be dealt with in the same framework. However, naive realisations may have a prohibitive running time due to the large number of correlated latent variables, and more advanced techniques can be difficult to handle in practice. While MCMC is maybe the most advanced and widely used class of approximate inference techniques, it is not discussed in any further detail in this thesis.

2.1.2.1 Generalised Linear Models. Binary Classification

A large class of models of this kind is obtained by starting from *generalised linear models (GLMs)* [133, 118] and replacing the parametric linear function $\boldsymbol{x}^T\boldsymbol{\beta}$ by

[14] This is generalised easily to allow for bounded linear functionals of the latent process $u(\cdot)$ instead of the evaluation functional $\delta_{\boldsymbol{x}_i}$, as discussed in Section 2.1.4.

a process $u(\boldsymbol{x})$ with a GP prior. This can be seen as direct infinite-dimensional generalisation of GLMs by employing the weight space view (see Section 2.1.1). In the spline smoothing context, this framework is presented in detail in [66]. It employs noise distributions

$$P(y|u) = \exp\left(\phi^{-1}(y\,u - b(u)) + c(y,\phi)\right),$$

i.e. $P(y|u)$ is in an exponential family with natural parameter u, sufficient statistics y/ϕ and log partition function $\phi^{-1}b(u)$ (see Section A.4.1). Here, $\phi > 0$ is a scale hyperparameter. The linear model is a special case with $\phi = \sigma^2$, $u = \mu = \mathrm{E}_u[y]$ and $b(u) = (1/2)u^2$. A technically attractive feature of this framework is that $\log P(y|u)$ is strictly concave in u, leading to a strictly concave, unimodal posterior $P(\boldsymbol{u}|S)$. For binary classification and $y \in \{-1, +1\}$, the GLM for the binomial noise distribution is *logistic regression* with the *logit* noise

$$P(y|u) = \sigma(y\,(u+b)),\ \sigma(t) = \frac{1}{1+e^{-t}}. \tag{2.8}$$

Another frequently used binary classification noise model is *probit* noise

$$P(y|u) = \Phi(y\,(u+b)) = \mathrm{E}_{\tau\sim N(0,1)}\left[\mathrm{I}_{\{y(u+b)+\tau>0\}}\right] \tag{2.9}$$

which can be seen as noisy Heaviside step. Both noise models are strictly log-concave.

2.1.2.2 Models with C Latent Processes

We can also allow for a fixed number $C \geq 1$ of latent variables for each case $(\boldsymbol{x}, \boldsymbol{y})$, i.e. C processes $u_c(\boldsymbol{x})$. The likelihood factors as

$$\prod_{i=1}^{n} P(\boldsymbol{y}_i|\boldsymbol{u}^{(i)}), \quad \boldsymbol{u}^{(i)} = (u_c(\boldsymbol{x}_i))_c.$$

$u_c(\boldsymbol{x})$ is zero-mean Gaussian *a priori* with covariance function $K^{(c)}$. While it is theoretically possible to use cross-covariance functions for prior covariances between u_c for different c, it may be hard to come up with a suitable class of such functions. Furthermore, the assumption that the processes u_c are independent

a priori leads to large computational savings, since the joint covariance matrix over the data assumes block-diagonal structure. Note that in this structure, we separate w.r.t. different c, while in block-diagonal structures coming from the factorised likelihood we separate w.r.t. cases i.

An important example using C latent processes is C-class classification. The likelihood comes from a multinomial GLM (or multiple logistic regression). It is convenient to use a binary encoding for the class labels, i.e. $\boldsymbol{y} = \boldsymbol{\delta}_c$ for class $c \in \{1, \ldots, C\}$.[15] The noise is multinomial with

$$\boldsymbol{\mu} = \mathrm{E}[\boldsymbol{y} \mid \boldsymbol{u}] = \mathrm{softmax}(\boldsymbol{u}) = \left(\mathbf{1}^T \exp(\boldsymbol{u})\right)^{-1} \exp(\boldsymbol{u}).$$

$\boldsymbol{u} \mapsto \boldsymbol{\mu}$ is sometimes called *softmax* mapping. Note that this mapping is not invertible, since we can add $\alpha\mathbf{1}$ to $\boldsymbol{u}$ for any α without changing $\boldsymbol{\mu}$. In terms of Section A.4.1, the parameterisation of the multinomial by $\boldsymbol{u}$ is overcomplete, due to the linear constraint $\boldsymbol{y}^T\mathbf{1} = 1$ on $\boldsymbol{y}$, and the corresponding GLM log partition function

$$b(\boldsymbol{u}) = \log \mathbf{1}^T \exp(\boldsymbol{u})$$

is not strictly convex. The usual remedy is to constrain $\boldsymbol{u}$ by for example fixing $u_C = 0$. This is fine in the context of fitting parameters by ML, but may be problematic for Bayesian inference. As mentioned above, we typically use priors which are i.i.d. over the u_c, so if we fix $u_C = 0$, the induced prior on $\boldsymbol{\mu}$ is not an exchangeable distribution (i.e. component permutations of $\boldsymbol{u}$ can have different distributions) and μ_C is singled out for no other than technical reasons. We think it is preferable in the Bayesian context to retain symmetry and accept that $\boldsymbol{u} \mapsto \boldsymbol{\mu}$ is not 1-to-1. Dealing with this non-identifiability during inference approximations is not too hard since softmax is invertible on any plane orthogonal to $\mathbf{1}$ and $b(\boldsymbol{u})$ is strictly convex on such. Anyway, this detail together with the two different blocking structures mentioned above renders implementations of approximate inference for the C-class model somewhat more involved than the binary case. Furthermore, it turns out that the straightforward extension of inference approximations used in this thesis (e.g., Sections 4.3 and 4.4) grows by

[15] We use vector notation for $\boldsymbol{u}, \boldsymbol{y} \in \mathbb{R}^C$ associated with a single case. This should not be confused with the vector notation $\boldsymbol{u}, \boldsymbol{y} \in \mathbb{R}^n$ used above to group variables for all cases.

a factor of C^2 in running time requirements compared to the binary case. In contrast, the GP Laplace approximation (see Section 2.1.3) scales linearly in C only. It is possible that by using further approximations, such a desirable scaling can be attained for the former methods as well, but this is not pursued in this thesis. Other examples for C-process models are ordinal regression ("ranking") models (see [118] for likelihood suggestions) or multivariate regression.

Note that it is straightforward to fit a C-class model to uncertain class observations. For example, if the target $\boldsymbol{y}$ is a distribution over $1, \ldots, C$, the corresponding log likelihood factor can be chosen as

$$\log P(\boldsymbol{y}|\boldsymbol{u}) = -\mathrm{D}[\boldsymbol{y} \,\|\, \boldsymbol{\mu}], \quad \boldsymbol{\mu} = \mathrm{softmax}(\boldsymbol{u}),$$

which reduces to normal case for $\boldsymbol{y} = \boldsymbol{\delta}_c$. Note that this is concave in $\boldsymbol{u}$, and strictly concave on each plane orthogonal to $\mathbf{1}$. This model is also useful for problems in which points $\boldsymbol{x}$ can belong to multiple classes (multi-label problems).

2.1.2.3 Robust Regression

GP regression with Gaussian noise can lead to poor results if the data is prone to outliers, due to the light tails of the noise distribution. A robust GP regression model can be obtained by using a heavy-tailed noise distribution $P(y|u)$ such as a Laplace or even Student-t distribution. An interesting idea is to use the fact that the latter is obtained by starting with $N(0, \tau^{-1})$ and to integrate out the precision τ over a Gamma distribution (e.g., [131]). Thus, a robust model can be written as

$$y = u + \varepsilon, \quad \varepsilon \sim N(0, \tau^{-1}),$$

where τ is drawn i.i.d. from a Gamma distribution (whose parameters are hyperparameters). The posterior $P(\boldsymbol{u}|S, \boldsymbol{\tau})$ conditioned on the precision values τ_i is Gaussian and is computed in the same way as for the case $\tau_i = \sigma^{-2}$ above. $\boldsymbol{\tau}$ can be sampled by MCMC, or may be chosen to maximise the posterior $P(\boldsymbol{\tau}|S)$. The marginal likelihood $P(\boldsymbol{y}|\boldsymbol{\tau})$ is Gaussian and can be computed easily. However, note that in the latter case the number of hyperparameters grows as n which

might invalidate the usual justification of marginal likelihood maximisation (see Section A.6.2).

2.1.3 Approximate Inference and Learning

We have seen in the previous section that the posterior process for a likelihood of the general form (2.6) can be written as "shifted" version (2.7) of the prior. About the only processes (in this context) which can be dealt with feasibly are Gaussian ones, and a general way of obtaining a GP approximation to the posterior process is to approximate $P(\boldsymbol{u}|S)$ by a Gaussian $Q(\boldsymbol{u}|S)$, leading to the process

$$dQ(u(\cdot)) = \frac{Q(\boldsymbol{u}|S)}{P(\boldsymbol{u})} dP(u(\cdot)) \tag{2.10}$$

which is Gaussian. An optimal way of choosing Q would be to minimise

$$\mathrm{D}[P(u(\cdot)|S) \,\|\, Q(u(\cdot))] = \mathrm{D}[P(\boldsymbol{u}|S) \,\|\, Q(\boldsymbol{u}|S)]. \tag{2.11}$$

The equality follows from the fact that if $dP(u(\cdot)|S) \ll dQ(u(\cdot)|S)$, then

$$dP(u(\cdot)|S) = \frac{P(\boldsymbol{u}|S)}{Q(\boldsymbol{u}|S)}\, dQ(u(\cdot)|S),$$

and otherwise $\mathrm{D}[P(\boldsymbol{u}|S) \,\|\, Q(\boldsymbol{u}|S)] = \infty$ (recall our notation from Section A.1). At the minimum point (unique w.r.t. f.d.d.'s of Q) Q and P have the same mean and covariance function. This is equivalent to moment matching (see Section A.4.2) and requires us to find mean and covariance matrix of $P(\boldsymbol{u}|S)$. Unfortunately, this is intractable in general for large datasets and non-Gaussian noise. Any other Gaussian approximation $Q(\boldsymbol{u}|S)$ leads to a GP posterior approximation $Q(u(\cdot))$, and the intractable (2.11) can nevertheless be valuable as guideline, as for example in the EP algorithm (see Section C.1.1). In this thesis, we are primarily interested in approximate inference methods for GP models which employ GP approximations (2.10) to posterior processes via

$$Q(\boldsymbol{u}|S) = N(\boldsymbol{u} \,|\, \boldsymbol{K}\boldsymbol{\xi}, \boldsymbol{A}). \tag{2.12}$$

Here, $\boldsymbol{\xi}$, $\boldsymbol{A}$ can depend on the data S, the covariance function K (often via the kernel matrix $\boldsymbol{K}$) and on other hyperparameters. This class contains a variety of

methods proposed in the literature, some of them are briefly discussed below (see also Section 4.3). Virtually all of these have a reduced $O(n)$ parameterisation, since $\boldsymbol{A}$ has the restricted form

$$\boldsymbol{A} = \left(\boldsymbol{K}^{-1} + \boldsymbol{I}_{\cdot,I}\boldsymbol{D}\boldsymbol{I}_{I,\cdot}\right)^{-1} \tag{2.13}$$

with $\boldsymbol{D} \in \mathbb{R}^{d,d}$ diagonal with positive entries and $I \subset \{1,\ldots,n\}$, $|I| = d$ (recall our notation $\boldsymbol{I}_{\cdot,I}$ from Section A.1). For the methods mentioned below in this section, $d = n$ and $\boldsymbol{I}_{\cdot,I} = \boldsymbol{I}$, but for the sparse approximations in Chapter 4 we have $d \ll n$. In the latter case, $\boldsymbol{\xi}_{\setminus I} = \boldsymbol{0}$ and we use $\boldsymbol{\xi} \in \mathbb{R}^d$ for simplicity, replacing $\boldsymbol{\xi}$ in (2.12) by $\boldsymbol{I}_{\cdot,I}\boldsymbol{\xi}$.

From (2.10), the (approximate) predictive posterior distribution of $u_* = u(\boldsymbol{x}_*)$ at a test point $\boldsymbol{x}_*$ is determined easily as $Q(u_*|\boldsymbol{x}_*, S) = N(u_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$, where

$$\begin{aligned} \mu(\boldsymbol{x}_*) &= \boldsymbol{k}_I(\boldsymbol{x}_*)^T\boldsymbol{\xi}, \\ \sigma^2(\boldsymbol{x}_*) &= K(\boldsymbol{x}_*,\boldsymbol{x}_*) - \boldsymbol{k}_I(\boldsymbol{x}_*)^T\boldsymbol{D}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{D}^{1/2}\boldsymbol{k}_I(\boldsymbol{x}_*), \\ \boldsymbol{B} &= \boldsymbol{I} + \boldsymbol{D}^{1/2}\boldsymbol{K}_I\boldsymbol{D}^{1/2}. \end{aligned} \tag{2.14}$$

Here, $\boldsymbol{k}_I(\boldsymbol{x}_*) = (K(\boldsymbol{x}_i,\boldsymbol{x}_*))_{i\in I}$. More general, the GP posterior approximation has mean function $\mu(\boldsymbol{x})$ and covariance function

$$K(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{k}_I(\boldsymbol{x})^T\boldsymbol{D}^{1/2}\boldsymbol{B}^{-1}\boldsymbol{D}^{1/2}\boldsymbol{k}_I(\boldsymbol{x}').$$

The predictive distribution $P(y_*|\boldsymbol{x}_*, S)$ is obtained by averaging $P(y_*|u_*)$ over $N(u_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$. If this expectation is not analytically tractable, it can be done by Gaussian quadrature (see Section C.1.2) if $P(y_*|u_*)$ is smooth and does not grow faster than polynomial. A simple and numerically stable way to determine the predictive variances is to compute the Cholesky decomposition $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T$ (Definition A.2) after which each variance requires one back-substitution with $\boldsymbol{L}$. We will refer to $\boldsymbol{\xi}$ as *prediction vector*. More generally, as mentioned in Section 2.1.1, we can use derivative information or other bounded linear functionals of the latent process $u(\boldsymbol{x})$ in the likelihood and/or for the variables to be predicted, using the fact that the corresponding finite set of scalar

variables is multivariate Gaussian with prior covariance matrix derived from the covariance function K (as discussed in more detail in Section 2.1.4).

A generalisation to the multi-process models of Section 2.1.2 is also straightforward in principle. Here, $\boldsymbol{u}$ has dimension $C\,n$. Again $\boldsymbol{A}$ is restricted to the form (2.13), although $\boldsymbol{D}$ is merely block-diagonal with n $(d \times d)$ blocks on the diagonal. Moreover, if the processes are *a priori* independent, both $\boldsymbol{K}$ and $\boldsymbol{K}^{-1}$ consist of C $(n \times n)$ blocks on the diagonal. The general formulae for prediction (2.14) have to be modified for efficiency. The details are slightly involved and may depend on the concrete approximation method, C-process models are not discussed in further detail in this thesis.

2.1.3.1 Some Examples

A simple and efficient way of obtaining a Gaussian approximation $Q(\boldsymbol{u}|S)$ is via Laplace's method (see Section A.6.3), as proposed in [210] for binary classification with logit noise (2.8). To this end, we have to find the posterior mode $\hat{\boldsymbol{u}}$ which can be done by a variant of Newton-Raphson (or Fisher scoring, see [118]). Each iteration consists of a weighted regression problem, i.e. requires the solution of an $n \times n$ positive definite linear system. This can be done approximately in $O(n^2)$ using a conjugate gradients solver. At the mode, we have

$$\boldsymbol{\xi} = \boldsymbol{Y}\,\sigma(-\boldsymbol{Y}\,\hat{\boldsymbol{u}}), \quad \boldsymbol{D} = (\operatorname{diag}\sigma(-\boldsymbol{Y}\,\hat{\boldsymbol{u}}))(\operatorname{diag}\sigma(\boldsymbol{Y}\,\hat{\boldsymbol{u}})), \tag{2.15}$$

where σ is the logistic function (2.8) and $\boldsymbol{Y} = \operatorname{diag}\boldsymbol{y}$. All n diagonal elements of $\boldsymbol{D}$ are positive. Recall that the Laplace approximation replaces the log posterior by a quadratic fitted to the local curvature at the mode $\hat{\boldsymbol{u}}$. For the logit noise the log posterior is strictly concave and dominated by the Gaussian prior far out, so in general a Gaussian approximation should be fairly accurate. On the other hand, the true posterior can be significantly skewed, meaning that the mode can be quite distant from the mean (which would be optimal) and the covariance approximation via local curvature around the mode can be poor.

The expectation propagation (EP) algorithm for GP models is described in Section 4.3 and employed for sparse approximations in Chapter 4. It can signif-

icantly outperform the Laplace GP approximation in terms of prediction accuracy (e.g., [139]), but is also more costly to train.[16] It is also somewhat harder to ensure numerical stability. On the other hand, EP is more general and can for example deal with discontinuous or non-differentiable log likelihoods. Our description of EP applies to C-process models just as well, with the unfortunate C^2 scaling and the slight technical difficulty of having to approximate C-dimensional Gaussian expectations (see Section C.1.2 for some references). A range of different variational approximations (see Section A.6.3) have been suggested in [63, 168, 82]. Note that for the variational method where $Q(\boldsymbol{u}|S)$ is chosen to minimise $\mathrm{D}[\cdot \,\|\, P(\boldsymbol{u}|S)]$, it is easy to see that the best Gaussian variational distribution has a covariance matrix of the form (2.13).

All approximations mentioned so far have training time scaling of $O(n^3)$ which is prohibitive for large datasets. Sparse inference approximations reduce this scaling to $O(n\,d^2)$ with controllable $d \ll n$ and are the topic of Chapter 4. In contrast to this, at least for fairly low-dimensional input spaces $\mathcal{X}$ (e.g., $\mathbb{R}$, $\mathbb{R}^2$ or the 2-dimensional surface of a sphere) spline smoothing methods are very attractive for computational reasons. These methods are discussed in some detail in Section 2.1.5. In a nutshell, they enforce smoothness by penalising curvature or other derivatives of low order. For the corresponding spline kernel as covariance function of a GP prior, predictive posterior GPs are spline functions (i.e. piecewise polynomial), and the linear systems during training can be solved in $O(n)$ due to band-structured system matrices.

2.1.3.2 Model Selection

So far we have only been concerned with first-level inference conditioned on fixed hyperparameters. A useful general method has to provide some means to select good values for these parameters or to marginalise over them (see Section 2.1.2). In this thesis, we will focus on marginal likelihood maximisation as general model selection technique (see Section A.6.2). If we denote the hyperparameters by $\boldsymbol{\alpha}$,

[16] Partly due to its more complex iterative structure, but also because its elementary steps are smaller than for the Laplace technique and cannot be vectorised as efficiently.

the log marginal likelihood $\log P(\boldsymbol{y}|\boldsymbol{\alpha})$ is as difficult to compute as the posterior $P(\boldsymbol{u}|S,\boldsymbol{\alpha})$ and has to be approximated in general.[17] We use the variational lower bound described in Section A.6.4, thus employ lower bound maximisation for model selection. Namely, in our case the lower bound (A.19) becomes

$$\begin{aligned}\log P(\boldsymbol{y}|\boldsymbol{\alpha}) &\geq \mathrm{E}_Q\left[\log P(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{\alpha}) + \log P(\boldsymbol{u}|\boldsymbol{\alpha})\right] + \mathrm{H}[Q(\boldsymbol{u}|S)] \\ &= \mathrm{E}_Q\left[\log P(\boldsymbol{y}|\boldsymbol{u},\boldsymbol{\alpha})\right] - \mathrm{D}[Q(\boldsymbol{u}|S) \,\|\, P(\boldsymbol{u}|\boldsymbol{\alpha})].\end{aligned} \tag{2.16}$$

(the differential entropy $\mathrm{H}[\cdot]$ and relative entropy $\mathrm{D}[\cdot]$ are defined in Section A.3). Note that the posterior approximation $Q(\boldsymbol{u}|S)$ depends on $\boldsymbol{\alpha}$ as well, but it is not feasible in general to obtain its exact gradient w.r.t. $\boldsymbol{\alpha}$. Variational EM, an important special case of lower bound maximisation is iterative, in turn freezing one of Q, $\boldsymbol{\alpha}$ and maximising the lower bound w.r.t. the other (here, Q can be chosen from a family of variational distributions). Alternatively, Q can be chosen in a different way as approximation of the posterior $P(\boldsymbol{u}|S)$. The deviation from the variational choice of Q can be criticised on the ground that other choices of Q can lead to decreases in the lower bound, so the overall algorithm does not increase its criterion strictly monotonically. On the other hand, Q chosen in a different way may lie outside families over which the lower bound can be maximised efficiently, thus may even result in a larger value than the family maximiser.[18] Furthermore, the lower bound criterion can be motivated by the fact that its gradient

$$\mathrm{E}_{Q(\boldsymbol{u}|S)}\left[\nabla_{\boldsymbol{\alpha}} \log P(\boldsymbol{y},\boldsymbol{u}|\boldsymbol{\alpha})\right]$$

(ignoring the dependence of Q on $\boldsymbol{\alpha}$) approximates the true gradient

$$\nabla_{\boldsymbol{\alpha}} \log P(\boldsymbol{y}|\boldsymbol{\alpha}) = \mathrm{E}_{P(\boldsymbol{u}|S)}\left[\nabla_{\boldsymbol{\alpha}} \log P(\boldsymbol{y},\boldsymbol{u}|\boldsymbol{\alpha})\right]$$

at every point $\boldsymbol{\alpha}$ (see Section A.6.4). In the context of approximate GP inference methods, the dependence of $Q(\boldsymbol{u}|S)$ on the GP prior (thus on $\boldsymbol{\alpha}$) is quite explicit (for example, the covariance of Q is $(\boldsymbol{K}^{-1} + \boldsymbol{D})^{-1}$ which depends strongly on the

[17] It is analytically tractable for a Gaussian likelihood, for example in the case of GP regression with Gaussian noise discussed above it is $\log N(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K} + \sigma^2\boldsymbol{I})$.

[18] For example, even though the bound maximiser over all Gaussians has a covariance matrix of the form (2.13), finding it is prohibitively costly in practice and proposed variational schemes [63, 168, 82] use restricted subfamilies.

kernel matrix $\boldsymbol{K}$). We argue that instead of keeping all of Q fixed for the gradient computation, we should merely ignore the dependence of the essential parameters $\boldsymbol{\xi}$, $\boldsymbol{D}$ on $\boldsymbol{\alpha}$. This typically leads to a more involved gradient computation which is potentially closer to the true gradient. Alternatively, if this computation is beyond resource limits, further indirect dependencies on $\boldsymbol{\alpha}$ may be ignored. We refer to Section C.3.3 for a concrete example and to Section 4.5.3 for details about the optimisation problem which is slightly non-standard due to the lack of strict monotonicity.

2.1.4 Reproducing Kernel Hilbert Spaces

The theory of *reproducing kernel Hilbert spaces (RKHS)* can be used to characterise the space of random variables obtained as bounded linear functionals of a GP on which any method of prediction from finite information must be based. Apart from that, RKHS provide a unification of ideas from a wide area of mathematics, most of which will not be mentioned here. The interested reader may consult [6]. Our exposition is taken from [200]. None of the theory discussed here is used for the essential arguments in this thesis, and the section can be skipped by readers not interested in the details.

A *reproducing kernel Hilbert space (RKHS)* $\mathcal{H}$ is a Hilbert space of functions $\mathcal{X} \to \mathbb{R}$ for which all evaluation functionals $\delta_{\boldsymbol{x}}$ are bounded. This implies that there exists a kernel $K(\boldsymbol{x}, \boldsymbol{x}')$ s.t. $K(\cdot, \boldsymbol{x}) \in \mathcal{H}$ for all $\boldsymbol{x} \in \mathcal{X}$ and

$$f(\boldsymbol{x}) = \delta_{\boldsymbol{x}} f = (K(\cdot, \boldsymbol{x}), f) \tag{2.17}$$

for all $f \in \mathcal{H}$, where $(\cdot, \cdot)$ is the inner product in $\mathcal{H}$. To be specific, a Hilbert space is a vector space with an inner product which is complete, in the sense that each Cauchy sequence converges to an element of the space. For example, a Hilbert space $\mathcal{H}$ can be generated from an inner product space of functions $\mathcal{X} \to \mathbb{R}$ by adjoining the limits of all Cauchy sequences to $\mathcal{H}$. Note that this is a rather abstract operation and the adjoined objects need not be functions in the usual sense: for example $\mathcal{L}_2(\mu)$, the Hilbert space of functions for which

$$\int f(\boldsymbol{x})^2 \, d\mu(\boldsymbol{x}) < \infty \tag{2.18}$$

contains the Dirac delta "functions" $\delta_{\boldsymbol{x}}$ (representers of the evaluation functionals) which are not pointwise defined. For an RKHS $\mathcal{H}$ such anomalies cannot occur, since the functionals $\delta_{\boldsymbol{x}}$ are bounded:[19]

$$|f(\boldsymbol{x})| = |\delta_{\boldsymbol{x}} f| \leq C_{\boldsymbol{x}} \|f\|.$$

By the Riesz representation theorem, there exists a unique *representer* $K_{\boldsymbol{x}} \in \mathcal{H}$ such that (2.17) holds with $K(\cdot, \boldsymbol{x}) = K_{\boldsymbol{x}}$. It is easy to see that the kernel K is positive semidefinite. K is called *reproducing kernel (RK)* of $\mathcal{H}$, note that

$$\left(K_{\boldsymbol{x}}, K_{\boldsymbol{x}'}\right) = (K(\cdot, \boldsymbol{x}), K(\cdot, \boldsymbol{x}')) = K(\boldsymbol{x}, \boldsymbol{x}').$$

It is important to note that in a RKHS, (norm) convergence implies pointwise convergence to a pointwise defined function, since

$$|f_m(\boldsymbol{x}) - f(\boldsymbol{x})| = |(K_{\boldsymbol{x}}, f_m - f)| \leq C_{\boldsymbol{x}} \|f_m - f\|.$$

On the other hand, for any positive semidefinite K there exists a unique RKHS $\mathcal{H}$ with RK K. Namely, the set of finite linear combinations of $K(\cdot, \boldsymbol{x}_i)$, $\boldsymbol{x}_i \in \mathcal{X}$ with

$$\left(\sum_i a_i K(\cdot, \boldsymbol{x}_i), \sum_j b_j K(\cdot, \boldsymbol{x}'_j)\right) = \sum_{i,j} a_i b_j K(\boldsymbol{x}_i, \boldsymbol{x}'_j)$$

is an inner product space which is extended to a Hilbert space $\mathcal{H}$ by adjoining all limits of Cauchy sequences. Since norm convergence implies pointwise convergence in the inner product space, all adjoined limits are pointwise defined functions and $\mathcal{H}$ is an RKHS with RK K. To conclude, a RKHS has properties which make it much "nicer" to work with than a general Hilbert space. All functions are pointwise defined, and the representer of the evaluation functional $\delta_{\boldsymbol{x}}$ is explicitly given by $K(\cdot, \boldsymbol{x})$.

2.1.4.1 RKHS by Mercer Eigendecomposition. Karhunen-Loeve Expansion

We have already seen that $\mathcal{L}_2(\mu)$ is not a RKHS in general, but for many kernels K it contains a (unique) RKHS as subspace. Recall that $\mathcal{L}_2(\mu)$ contains all

[19] Bounded functionals are sometimes called *continuous*.

functions $f : \mathcal{X} \to \mathbb{R}$ for which (2.18) holds. The standard inner product is

$$(f, g) = \int f(\boldsymbol{x}) g(\boldsymbol{x})\, d\mu(\boldsymbol{x}).$$

Often, μ is taken as indicator function of a compact set such as the unit cube. A positive semidefinite $K(\boldsymbol{x}, \boldsymbol{x}')$ can be regarded as kernel (or representer) of a positive semidefinite linear operator $\mathcal{K}$ in the sense

$$(\mathcal{K}f)(\boldsymbol{x}) = (K(\cdot, \boldsymbol{x}), f).$$

ϕ is an eigenfunction of K with eigenvalue $\lambda \neq 0$ if

$$(\mathcal{K}\phi)(\boldsymbol{x}) = (K(\cdot, \boldsymbol{x}), \phi) = \lambda\, \phi(\boldsymbol{x})$$

for all $\boldsymbol{x}$. For K, all eigenvalues are real and non-negative. Furthermore, suppose K is continuous and

$$\int K(\boldsymbol{x}, \boldsymbol{x}')^2\, d\mu(\boldsymbol{x}) d\mu(\boldsymbol{x}') < \infty.$$

Then, by the Mercer-Hilbert-Schmidt theorems there exists a countable[20] orthonormal sequence of continuous eigenfunctions $\phi_\nu \in \mathcal{L}_2(\mu)$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$, and K can be expanded in terms of these:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\nu \geq 1} \lambda_\nu \phi_\nu(\boldsymbol{x}) \phi_\nu(\boldsymbol{x}'), \tag{2.19}$$

and $\sum_{\nu \geq 1} \lambda_\nu^2 < \infty$, thus $\lambda_\nu \to 0 (\nu \to \infty)$. This can be seen as generalisation of the eigendecomposition of a positive semidefinite Hermitian matrix. Indeed, the reproducing property of positive semidefinite kernels was recognised and used by E. H. Moore [126] to develop the notion of general "positive Hermitian matrices". In this case, we can characterise the RKHS embedded in $\mathcal{L}_2(\mu)$ explicitly. For $f \in \mathcal{L}_2(\mu)$, define the Fourier coefficients

$$f_\nu = (f, \phi_\nu).$$

[20] Such operators (kernels) are called *trace-class*, and in general only such will have a discrete spectrum (countable number of eigenvalues).

Consider the subspace $\mathcal{H}_K$ of all $f \in \mathcal{L}_2(\mu)$ with $\sum_{\nu\geq 1} \lambda_\nu^{-1} f_\nu^2 < \infty$. Then, $\mathcal{H}_K$ is a Hilbert space with inner product

$$(f, g)_K = \sum_{\nu\geq 1} \frac{f_\nu g_\nu}{\lambda_\nu},$$

moreover the Fourier series $\sum_{\nu\geq 1} f_\nu \phi_\nu$ converges pointwise to f. Since $\{\lambda_\nu \phi_\nu(\boldsymbol{x})\}$ are the Fourier coefficients of $K(\cdot, \boldsymbol{x})$ (using Equation 2.19), we have

$$(f, K(\cdot, \boldsymbol{x}))_K = \sum_{\nu\geq 1} f_\nu \phi_\nu(\boldsymbol{x}) = f(\boldsymbol{x}),$$

thus K is the RK of $\mathcal{H}_K$. It is important to distinguish clearly between the inner products $(\cdot, \cdot)$ in $\mathcal{L}_2(\mu)$ and $(\cdot, \cdot)_K$ in $\mathcal{H}_K$ (see [216] for more details about the relationship of these inner products). While $\|\cdot\|$ measures "expected squared distance" from 0 (w.r.t. $d\mu$), $\|\cdot\|_K$ is a measure of the "roughness" of a function. For example, the eigenfunctions have $\|\phi_\nu\| = 1$, but $\|\phi_\nu\|_K = \lambda_\nu^{-1/2}$ thus becoming increasingly rough.[21]

The spectral decomposition of K leads to an important representation of a zero-mean GP $u(\boldsymbol{x})$ with covariance function K: the *Karhunen-Loeve expansion*. Namely, the sequence

$$u_k(\boldsymbol{x}) = \sum_{\nu=1}^{k} u_\nu \phi_\nu(\boldsymbol{x}), \tag{2.20}$$

where u_ν are independent $N(0, \lambda_\nu)$ variables, converges to $u(\boldsymbol{x})$ in quadratic mean (a stronger statement under additional conditions can be found in [2]). Moreover,

$$u_\nu = \int u(\boldsymbol{x}) \phi_\nu(\boldsymbol{x})\, d\mu(\boldsymbol{x})$$

which is well defined in quadratic mean. We have already used this expansion in Section 2.1.1 to introduce the "weight space view". Note that since the variances λ_ν decay to 0, the GP can be approximated by finite partial sums of the expansion (see [216]).

[21] In the same sense as high-frequency components in the usual Fourier transform.

2.1.4.2 Duality between RKHS and Gaussian Process

If $u(\boldsymbol{x})$ is a zero-mean GP with covariance function K, what is the exact relationship between $u(\boldsymbol{x})$ and the RKHS with RK K? One might think that $u(\boldsymbol{x})$ can be seen as distribution over $\mathcal{H}_K$, but this is wrong (as pointed out in [200], Sect. 1.1). In fact, for any version of $u(\boldsymbol{x})$ sample functions from the process are *not* in $\mathcal{H}_K$ with probability 1! This can be seen by noting that for the partial sums (2.20) we have

$$\mathrm{E}\left[\|u_k\|_K^2\right] = \mathrm{E}\left[\sum_{\nu=1}^{k} \frac{u_\nu^2}{\lambda_\nu}\right] = k \to \infty \ (k \to \infty).$$

Roughly speaking, $\mathcal{H}_K$ contains "smooth", non-erratic functions from $\mathcal{L}_2(\mu)$, characteristics we cannot expect from sample paths of a random process. A better intuition about $\mathcal{H}_K$ is that it will turn out to contain expected values of $u(\boldsymbol{x})$ conditioned on a finite amount of information. An important duality between $\mathcal{H}_K$ and a Hilbert space based on $u(\boldsymbol{x})$ was noticed in [89]. Namely, construct a Hilbert space $\mathcal{H}_{GP}$ in the same way as above starting from positive semidefinite K, but replace $K(\cdot, \boldsymbol{x}_i)$ by $u(\boldsymbol{x}_i)$ and use the inner product

$$(A, B)_{GP} = \mathrm{E}[AB],$$

thus

$$\left(\sum_i a_i u(\boldsymbol{x}_i), \sum_j b_j u(\boldsymbol{x}'_j)\right)_{GP} = \sum_{i,j} a_i b_j K(\boldsymbol{x}_i, \boldsymbol{x}'_j).$$

$\mathcal{H}_{GP}$ is a space of random variables, not functions, but it is isometrically isomorphic to $\mathcal{H}_K$ under the mapping $u(\boldsymbol{x}_i) \mapsto K(\cdot, \boldsymbol{x}_i)$, with

$$(u(\boldsymbol{x}), u(\boldsymbol{x}'))_{GP} = \mathrm{E}[u(\boldsymbol{x})u(\boldsymbol{x}')] = K(\boldsymbol{x}, \boldsymbol{x}') = (K(\cdot, \boldsymbol{x}), K(\cdot, \boldsymbol{x}'))_K.$$

For most purposes, we can regard $\mathcal{H}_{GP}$ as RKHS with RK K. If L is a bounded linear functional on $\mathcal{H}_K$, it has a representer $\nu \in \mathcal{H}_K$ with $\nu(\boldsymbol{x}) = LK_{\boldsymbol{x}}$. The isometry maps ν to a random variable $Z \in \mathcal{H}_{GP}$ which we formally denote by $Lu(\cdot)$. Note that

$$\mathrm{E}\left[(Lu(\cdot))u(\boldsymbol{x})\right] = (\nu, K_{\boldsymbol{x}})_K = \nu(\boldsymbol{x}) = LK_{\boldsymbol{x}}.$$

More general, if $L^{(1)}, L^{(2)}$ are functionals with representers $\nu^{(1)}$, $\nu^{(2)}$ s.t. $\boldsymbol{x} \mapsto L^{(j)}K_{\boldsymbol{x}}$ are in $\mathcal{H}_K$, then

$$\mathrm{E}\left[(L^{(1)}u(\cdot))(L^{(2)}u(\cdot))\right] = (\nu^{(1)}, \nu^{(2)})_K = L^{(1)}_{\boldsymbol{x}}(K(\cdot, \boldsymbol{x}), \nu^{(2)})_K = L^{(1)}_{\boldsymbol{x}} L^{(2)}_{\boldsymbol{y}} K(\boldsymbol{x}, \boldsymbol{y}).$$

Again, it is clear that $Lu(\cdot)$ is (in general) very different from the process obtained by applying L to sample paths of $u(\boldsymbol{x})$. In fact, since the latter are almost surely not in $\mathcal{H}_K$, L does not even apply to them in general. The correct interpretation is in quadratic mean, using the isometry between $\mathcal{H}_{GP}$ and $\mathcal{H}_K$. As an example, suppose that $\mathcal{X} = \mathbb{R}^d$ and $L = D_{\boldsymbol{x}}$ is a differential functional evaluated at $\boldsymbol{x}$. Then, we retrieve the observations in Section 2.1.1 about derivatives of a GP. The space $\mathcal{H}_{GP}$ is of central importance for the sort of inference on GP models we are interested in, because it contains exactly the random variables we condition on or would like to predict in situations where only a finite amount of information is available (from observations which are linear functionals of the process).

2.1.5 Penalised Likelihood. Spline Smoothing

Historically, GP models originated from spline smoothing techniques via penalised likelihood estimation, and spline kernels are widely used due to the favourable approximation properties of splines and algorithmic advantages. A comprehensive account of spline smoothing and relations to Bayesian estimation in GP models is [200] which our exposition is mainly based on. Spline smoothing is a special case of penalised likelihood methods, giving another view on the reproducing kernel via the Green's function of a penalisation (or regularisation) operator which will be introduced below. This section is included mainly for illustrative purposes.

In Section 2.1.4 we have discussed the duality between a Gaussian process and the RKHS of its covariance function. Apart from the Bayesian viewpoint using GP models, a different and more direct approach to estimation in non-parametric models is the penalised likelihood approach, the oldest and most widely used incarnations of are spline smoothing methods. We will introduce the basic ideas for the one-dimensional model which leads to the general notion of regularisation operators, penalty functionals and their connections to RKHS. We omit all de-

tails, (important) computational issues and multidimensional generalisations, see [200] for details. A more elementary account is [66].

We will only sketch the ideas, for rigorous details see [200, 89]. Interpolation and smoothing by splines originates from the work of Schönberg [163]. A *natural spline* $s(x)$ of order m on $[0,1]$ is defined based on *knots* $0 < x_1 < \cdots < x_n < 1$. If π^k denotes the set of polynomials of order $\leq k$, then $s(x) \in \pi^{2m-1}$ on $[x_i, x_{i+1}]$, $s(x) \in \pi^{m-1}$ on $[0, x_1]$ and on $[x_n, 1]$, and $s \in C^{2m-2}$ overall. *Natural cubic splines* are obtained for $m = 2$. Define the *roughness penalty*

$$J_m(f) = \int_0^1 \left(f^{(m)}(x)\right)^2 dx.$$

$J_m(f)$ penalises large derivatives of order m by a large value, for example J_2 is large for functions of large curvature. Then, for some fixed function values the interpolant minimising $J_m(f)$ over all f for which the latter is defined[22] is a spline of order m. If we consider the related *smoothing* problem of minimising the penalised empirical risk

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \alpha J_m(f), \quad f \in \mathcal{W}_m[0,1], \tag{2.21}$$

it is clear that the minimiser is again a natural spline $s(x)$ of order m (any other $f \in \mathcal{W}_m[0,1]$ can be replaced by the spline with the same values at the knots, this does not change the risk term and cannot increase J_m). Now, from Taylor's theorem:

$$f(x) = \sum_{\nu=0}^{m-1} \frac{x^\nu}{\nu!} f^{(\nu)}(0) + \int_0^1 G_m(t,u) f^{(m)}(t)\, dt$$

with $G_m(t,u) = (t-u)_+^{m-1}/(m-1)!$ (here, $x_+ = x\mathrm{I}_{\{x \geq 0\}}$). If $f^{(\nu)}(0) = 0$, $\nu = 0, \ldots, m-1$ then $(G_m(t,\cdot), D^m f) = f(t)$, thus $G_m(t,u)$ is the *Green's function* for the boundary value problem $D^m f = g$. These functions f form a Hilbert space with inner product

$$(f,g)_K = \int_0^1 f^{(m)}(x) g^{(m)}(x)\, dx$$

[22] More precisely, over all $f \in \mathcal{W}_m[0,1]$, a so-called *Sobolev space* of all $f \in C^{m-1}[0,1]$ s.t. $f^{(m-1)}$ is absolutely continuous on $[0,1]$.

which is a RKHS with RK

$$K(s,t) = \int_0^1 G_m(s,u)G_m(t,u)\,du. \tag{2.22}$$

The boundary values can be satisfied by taking the direct sum of the space with π^{m-1}: the norm $\|\cdot\|_K$ is only a seminorm on this space because $\|p\|_K = 0$ for $p \in \pi^{m-1}$. In general (see [200, 144]), we make use of the duality between a RKHS and a *regularisation (pseudodifferential) operator* $\mathcal{P}$ on $\mathcal{L}_2(\mu)$. Let $\mathcal{H}$ be the Hilbert space of f s.t. $\mathcal{P}f \in \mathcal{L}_2(\mu)$. For $\mathcal{P}$, consider the pseudodifferential operator[23] $\mathcal{P}^*\mathcal{P}$. If this has a null space (such as π^{m-1} in the example above), we restrict $\mathcal{H}$ to the orthogonal complement. Now, the operator is positive definite and has an inverse (its *Green's function*) whose kernel K is the RK of $\mathcal{H}$. The inner product is

$$(f,g)_K = (\mathcal{P}f, \mathcal{P}g)$$

and the penalty functional is simply the squared RKHS norm. If $G(\boldsymbol{t},\boldsymbol{u})$ is s.t. $(G(\boldsymbol{t},\cdot),\mathcal{P}f) = f(\boldsymbol{t})$ for all $f \in \mathcal{H}$, the RK is given by

$$K(\boldsymbol{s},\boldsymbol{t}) = (G(\boldsymbol{s},\cdot), G(\boldsymbol{t},\cdot)).$$

On the other hand, we can start from an RKHS with RK K without null space and derive the corresponding regularisation operator $\mathcal{P}$. This can give additional insight into the meaning of a covariance function (see [144, 182]). In fact, if K is stationary and continuous, we can use Bochner's theorem (2.2). Namely, if $f(\boldsymbol{\omega})$ is the spectral density of K, we can take $f(\boldsymbol{\omega})^{-1/2}$ as spectrum of $\mathcal{P}$.[24] The one-dimensional example above is readily generalised to splines on the unit sphere or to *thin plate splines* in $\mathcal{X} = \mathbb{R}^d$, but the details get quite involved (see [200], Chap. 2).

Kimeldorf and Wahba [89] generalised this setup to a general variational problem in an RKHS, allowing for general bounded linear functionals $L_i f$ instead of $f(x_i)$ in (2.21). The minimiser is determined by $n + M$ coefficients, where M is the dimension of the null space of the RK K ($M = m + 1$ in the spline case

[23] $\mathcal{P}^*$ is the adjoint of $\mathcal{P}$, i.e. $(f,\mathcal{P}g) = (\mathcal{P}^*f, g)$.

[24] $\mathcal{P}$ is not uniquely defined, but only $\mathcal{P}^*\mathcal{P}$ (which has spectrum $f(\boldsymbol{\omega})^{-1}$).

above). These can be computed by direct formulae given in [200], Sect. 1.3. In the more general *penalised likelihood approach* [200, 66], n function values or linear functionals of f are used as latent variables of a likelihood (see Section 2.1.2), to obtain for example non-parametric extensions of GLMs [66]. The penalised likelihood is obtained by adding the penalty functional to the likelihood, and just as above the minimiser is determined by $n+M$ coefficients only (this *representer theorem* can be proved using the same argument as in the spline case above). In general, iterative methods are required to find values for these coefficients.

2.1.5.1 Bayesian View on Spline Smoothing

We close this section by reviewing the equivalence between spline smoothing and Bayesian estimation for a GP model pointed out by Kimeldorf and Wahba [89]. Given a positive semidefinite kernel K with M-dimensional null space, let the corresponding RKHS $\mathcal{H}$ be orthogonally decomposed into an M-dimensional space $\mathcal{H}_0$ with orthonormal basis p_ν and $\mathcal{H}_1$ s.t. K is positive definite on $\mathcal{H}_1$. Consider the model

$$F(\boldsymbol{x}) = \sum_{\nu=1}^{M} \theta_\nu p_\nu(\boldsymbol{x}) + b^{1/2} u(\boldsymbol{x}), \quad y_i = F(\boldsymbol{x}_i) + \varepsilon_i,$$

where $u(\boldsymbol{x})$ is a zero-mean GP with covariance function K and ε_i are independent $N(0, \sigma^2)$. Furthermore, $\boldsymbol{\theta} \sim N(\boldsymbol{0}, a\boldsymbol{I})$ *a priori*. On the other hand, let f_λ be the minimiser in $\mathcal{H}$ of the regularised risk functional

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2 + \lambda \|\mathcal{P}_1 f\|^2,$$

where $\mathcal{P}_1$ is the orthogonal projection onto $\mathcal{H}_1$. Kimeldorf and Wahba [89] show that f_λ lies in the span of $\{p_\nu \,|\, \nu = 1, \ldots, M\} \cup \{K(\cdot, \boldsymbol{x}_i) \,|\, i = 1, \ldots, n\}$ and give a numerical procedure for computing the coefficients. If we define $\hat{F}_a(\boldsymbol{x}) = \mathrm{E}[F(\boldsymbol{x}) \,|\, y_1, \ldots, y_n]$, then they show that

$$\lim_{a \to \infty} \hat{F}_a(\boldsymbol{x}) = f_\lambda(\boldsymbol{x}), \quad \lambda = \frac{\sigma^2}{n\, b}$$

for every fixed $\boldsymbol{x}$. The proof (see [200], Chap. 1) is a straightforward application of the duality between the RKHS $\mathcal{H}_1$ and the Hilbert space based on $u(\boldsymbol{x})$, as

described in Section 2.1.4. The procedure of dealing with $\mathcal{H}_0$ and the improper prior on $\boldsymbol{\theta}$ is awkward but is not necessary if the covariance function is positive definite (has null space $\{0\}$).

Finally, we note that a parametric extension of a non-parametric GP model can be sensible even if K is positive definite, leading to *semiparametric models* (or *partial splines*). For details about such models, we refer to [66], Chap. 4 and [200], Chap. 6.

2.1.6 Maximum Entropy Discrimination. Large Margin Classifiers

We regard GPs as building blocks for statistical models in much the same way as a parametric family of distributions (see Section 2.1.2 for examples). Statistical methods to estimate unknown parameters in such models follow different paradigms, and in machine learning the following have been most popular.

1. Probabilistic Bayesian paradigm:
   Has been introduced in Section 2.1.2. As noted in Section 2.1.3, the (intractable) posterior process is typically approximated by a GP itself.

2. Large margin (discriminative) paradigm:
   Here, a "posterior" process is obtained by associating margin constraints with observed data, then searching for a process which fulfils these (soft) constraints and at the same time is close to the prior GP, in a sense made concrete in this section. Since the constraints are linear in the latent outputs, the "posterior" process is always a GP with the same covariance as the prior.

The relationship between Bayesian methods and penalised likelihood or generalised spline smoothing methods has been discussed in Section 2.1.5. Large margin methods are special cases of spline smoothing models with a particular loss function which does not correspond to a probabilistic noise model (e.g., [201, 168, 187]). Several attempts have been made to phrase large margin discrim-

ination methods as approximations to Bayesian inference (e.g., [187, 168, 166]), but the paradigm separation suggested in [83] seems somewhat more convincing.

The connection between these two paradigms has been formulated in [83], this section is based on their exposition. The large margin paradigm has been made popular by the empirical success of the *support vector machine (SVM)* introduced below. In the Bayesian GP setting (see Section 2.1.2), the likelihood $P(\boldsymbol{y}|\boldsymbol{u})$ of the observed data $\boldsymbol{y}$ can be seen to impose "soft constraints" on the predictive distribution, in the sense that functions of significant volume under the posterior must not violate many of them strongly. In the large margin paradigm whose probabilistic view has been called *minimum relative entropy discrimination (MRED)* [83], such constraints are enforced more explicitly.[25] We introduce a set of latent *margin variables* $\boldsymbol{\gamma} = (\gamma_i)_i \in \mathbb{R}^n$, one for each datapoint. Along with GP prior $P(u(\cdot))$ on the latent function, we choose a prior $P(\boldsymbol{\gamma})$ over $\boldsymbol{\gamma}$. The margin prior encourages large margins $\gamma_i \gg 0$, as is discussed in detail below. The minimum relative entropy distribution $dQ(u(\cdot), \boldsymbol{\gamma})$ is defined as minimiser of $\mathrm{D}[Q \,\|\, P]$, subject to the *soft margin constraints*

$$E_{(u(\cdot),\boldsymbol{\gamma})\sim Q}\left[y_i u(\boldsymbol{x}_i) - \gamma_i\right] \geq 0, \ i = 1, \ldots, n. \tag{2.23}$$

Just as in the case of a likelihood function, these constraints depend on the values $\boldsymbol{u} = (u(\boldsymbol{x}_i))_i$ of the random process $u(\cdot)$ only. It is well known in information theory (e.g., [79], Sect. 3.1) that the solution to this constrained problem is given by

$$\frac{dQ(u(\cdot), \boldsymbol{\gamma})}{dP(u(\cdot), \boldsymbol{\gamma})} = Z(\boldsymbol{\lambda})^{-1} \exp\left(\sum_{i=1}^{n} \lambda_i \left(y_i u_i - \gamma_i\right)\right), \tag{2.24}$$

where

$$Z(\boldsymbol{\lambda}) = \mathrm{E}_{(u(\cdot),\boldsymbol{\gamma})\sim P}\left[\exp\left(\sum_{i=1}^{n} \lambda_i \left(y_i u_i - \gamma_i\right)\right)\right].$$

The value for the Lagrange multipliers $\boldsymbol{\lambda}$ is obtained by minimising the convex function $\log Z(\boldsymbol{\lambda})$ (sometimes called the *dual criterion*) under the constraints $\boldsymbol{\lambda} \geq \boldsymbol{0}$. Since the right hand side of (2.24) factorises between $u(\cdot)$ and $\boldsymbol{\gamma}$ and

[25] For notational simplicity, we do not use a bias term b here. The modifications to do so are straightforward. In the original SVM formulation, b can be seen to have a uniform (improper) prior.

the same holds for the prior P, we see that Q must factorise in the same way. Furthermore, it is immediate from (2.24) that $Q(u(\cdot))$ is again a Gaussian process with the same covariance kernel K as $P(u(\cdot))$ and with mean function $\mu(\boldsymbol{x}_*) = \boldsymbol{k}(\boldsymbol{x}_*)^T \boldsymbol{Y}\boldsymbol{\lambda}$, where $\boldsymbol{Y} = \operatorname{diag}(y_i)_i$. Due to the factorised form, we also have $Z(\boldsymbol{\lambda}) = Z_{u(\cdot)}(\boldsymbol{\lambda}) Z_{\boldsymbol{\gamma}}(\boldsymbol{\lambda})$ and

$$Z_{u(\cdot)}(\boldsymbol{\lambda}) = \mathrm{E}_{\boldsymbol{u}\sim P}\left[e^{\boldsymbol{\lambda}^T \boldsymbol{Y}\boldsymbol{u}}\right] = e^{\frac{1}{2}\boldsymbol{\lambda}^T \boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\lambda}}.$$

The form of $Z_{\boldsymbol{\gamma}}$ depends on the choice of the prior $P(\boldsymbol{\gamma})$ on the margin variables. Jaakkola et. al. [83] give some examples of such priors which encourage large margins. For example, if $P(\boldsymbol{\gamma}) = \prod_i P(\gamma_i)$, then $P(\gamma_i)$ should drop quickly for $\gamma_i < 1$ in order to penalise small and especially negative margins (empirical errors). In order for (2.23) to be a "soft constraint" only w.r.t. margin violations and also to mimic the SVM situation, we have to use $P(\gamma_i) = 0$ for $\gamma_i > 1$.[26] If $P(\gamma_i) \propto e^{-c(1-\gamma_i)} \mathrm{I}_{\{\gamma_i \leq 1\}}$, then

$$Z_{\boldsymbol{\gamma}}(\boldsymbol{\lambda}) \propto \prod_{i=1}^{n} \frac{e^{-\lambda_i}}{1-\lambda_i/c},$$

and the complete dual criterion is

$$\log Z(\boldsymbol{\lambda}) = -\sum_{i=1}^{n} (\lambda_i + \log(1-\lambda_i/c)) + \frac{1}{2}\boldsymbol{\lambda}^T \boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\lambda}, \quad \boldsymbol{\lambda} \geq \boldsymbol{0}. \tag{2.25}$$

Except for the potential term $\log(1-\lambda_i/c)$, this is identical to the SVM dual objective (see below). The so-called *hard margin SVM* for which margin constraints are enforced without allowing for violations, is obtained for $c \to \infty$. It converges only if the training data is indeed separable and is prone to over-complicated solutions. The effect of the potential term on the solution is limited (see [83]). It keeps λ_i from saturating to c exactly (which happens in SVM for misclassified patterns). The dual criterion can be optimised using efficient algorithms such as SMO [142], although the nonlinear potential term introduces minor complications.[27] Just like in SVM, sparsity in $\boldsymbol{\lambda}$ is encouraged and can be observed in practice.

[26] As in the SVM setup, the choice of 1 as margin width is arbitrary, because the distance can be re-scaled in terms of the prior variance.

[27] SMO makes use of the fact that the SVM criterion is quadratic with linear constraints.

To conclude, MRED gives a complete probabilistic interpretation of the SVM, or at least of a close approximation thereof. Note that SVM classification cannot be seen as MAP approximation to Bayesian inference for a probabilistic model, because its loss function does not correspond to a proper negative log likelihood [168, 137, 187]. Interestingly, the MRED view points out limitations of this framework as opposed to a Bayesian treatment of a Gaussian process model with a proper likelihood. Recall from above that the margin constraints are linear in the latent outputs $\boldsymbol{u}$, leading to the fact that the MRED "posterior" process $Q(u(\cdot))$ has the *same* covariance kernel K than the prior. While the constraints enforce the predictive mean to move from 0 *a priori* to $\mu(\boldsymbol{x})$, the predictive variances are simply the prior ones, independent of the data. At least to us this implies that if predictive variances (or error bars) are to be estimated besides simply performing a discrimination, then SVMs or other large margin discriminative methods are not appropriate (this point is discussed in more detail in Section 4.7.2).

More important is the lack of well-founded methods for model selection with SVM. For Bayesian GP methods, a general model selection strategy is detailed in Section 2.1.3. Alternatively, hyperparameters can be marginalised over approximately using MCMC techniques [132]. Model selection techniques for sparse Bayesian GP methods are discussed in Section 4.5. In contrast, model selection for SVM is typically done using variants of cross validation, which severely limits the number of free parameters that can be adapted.

2.1.6.1 The Support Vector Machine

For the sake of completeness we briefly review the SVM for binary classification. For details, see [161]. A detailed account on the history of SVM methods can be found there as well. The algorithm for binary classification has been proposed originally in [33]. SVM can be seen as special case of penalised risk minimisation or generalised smoothing splines (see Section 2.1.5), with the novelty that the risk functional is not continuous. This often leads to sparse predictors (in the sense that many of the n coefficients are 0), but also requires constrained optimisation for training.

Recall the weight space view from Section 2.1.1. For simplicity, we suppress the feature mapping here and write $\boldsymbol{x}$ instead of $\Phi(\boldsymbol{x})$, but the reader should keep in mind that linear functions discussed here may live in spaces of much higher dimension that the input space. If $u(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$ is a hyperplane, it separates all training data correctly if $y_i u(\boldsymbol{x}_i) > 0$ for all $i \in \{1, \ldots, n\}$. The distance of $\boldsymbol{x}_i$ from the hyperplane is $|u_i|/\|\boldsymbol{w}\|$ in the Euclidean space, so the hyperplane which maximises the minimal margin over the training set is the solution of

$$\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2, \quad \text{subj. to } \boldsymbol{Y}\boldsymbol{u} \geq \boldsymbol{1}.$$

This is the *hard-margin support vector machine* problem. Even if the data is separable, neglecting some points may lead to a larger margin. This is important especially in high-dimensional feature spaces where separations are typically possible, but may have very small margins due to outliers. The *soft-margin support vector machine* problem relaxes the constraints by introducing slack variables $\boldsymbol{\gamma} = (\gamma_i)_i$:

$$\min_{\boldsymbol{w},b,\boldsymbol{\gamma}} \|\boldsymbol{w}\|^2 + \boldsymbol{1}^T\boldsymbol{\gamma}, \quad \text{subj. to } \boldsymbol{Y}\boldsymbol{u} \geq \boldsymbol{1} - \boldsymbol{\gamma},\ \boldsymbol{\gamma} \geq \boldsymbol{0}.$$

The term $\boldsymbol{1}^T\boldsymbol{\gamma}$ penalises each slack value γ_i linearly.[28] By introducing Lagrange multipliers $\boldsymbol{\lambda} \geq \boldsymbol{0}$ for the margin constraints and $\boldsymbol{\beta} \geq \boldsymbol{0}$ for the nonnegativity of $\boldsymbol{\gamma}$ and minimising the Lagrangian w.r.t. the primal parameters $\boldsymbol{w}, b, \boldsymbol{\gamma}$, we obtain $\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{Y}\boldsymbol{\lambda}$, where the data matrix $\boldsymbol{X}$ contains the $\boldsymbol{x}_i$ as rows, furthermore the equality constraint $\boldsymbol{1}^T\boldsymbol{\lambda} = 0$. The dual variables $\boldsymbol{\beta}$ can be eliminated by adding upper bound constraints on $\boldsymbol{\lambda}$: $\lambda_i \in [0, 1]$. Recall from Lagrange multiplier theory that we can choose the dual parameters by maximising the Lagrangian, leading to the *dual* problem

$$\min_{\boldsymbol{\lambda}} \frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\lambda} - \boldsymbol{1}^T\boldsymbol{\lambda}, \quad \text{subj. to } \lambda_i \in [0, 1],\ i = \{1, \ldots, n\}.$$

The final discriminant function is $u(\boldsymbol{x}_*) = \boldsymbol{\lambda}^T\boldsymbol{Y}\boldsymbol{X}\boldsymbol{x}_* = \boldsymbol{\lambda}^T\boldsymbol{Y}\boldsymbol{k}(\boldsymbol{x}_*)$ with $\boldsymbol{k}(\boldsymbol{x}_*) = (K(\boldsymbol{x}_i, \boldsymbol{x}_*))_i$. The Lagrange multiplier theorem also renders the so-called *Karush-*

[28] Readers may miss the C parameter in front of the penalty, allowing to adjust the trade-off between it and $\|\boldsymbol{w}\|^2$. We prefer to re-scale the feature space inner product which is equivalent to multiplying the kernel K by C, the so-called variance parameter (see Section 2.1.8). In some applications it makes sense to use different penalty coefficients C_i for each of the γ_i, the required modifications are straightforward.

Kuhn-Tucker (KKT) conditions which must hold at the solution, namely whenever $\lambda_i \in (0,1)$, then the corresponding constraint must hold with equality: $y_i(u_i+b)=1$. The corresponding patterns are sometimes called *essential support vectors*. Also, it $y_i(u_i+b)<1$, i.e. $\gamma_i>0$, then we must have $\lambda_i=1$, these patterns are *bound support vectors*. Their union are the *support vectors*, i.e. the patterns whose dual variables in $\boldsymbol{\lambda}$ are non-zero. b can be determined from any set of essential support vectors. As mentioned above, the probabilistic MRED view on the SVM is not perfect due to the non-quadratic (yet convex) dual criterion, but certainly captures the essentials better than attempts to relate SVM to Bayesian MAP approximations.

We conclude this section by mentioning a well-known fact about SVM: it is a *compression scheme* in the sense defined in Section 3.5.4. In other words, once we have identified the set of support vectors, then retraining the machine on this reduced set only gives back the same solution. This follows from the fact that the KKT conditions for the reduced problem are the same as for the full problem with the non-SV dual coefficient set to 0. It also follows easily for MRED above, because if $\lambda_i=0$ it can simply be dropped from the criterion (2.25) (which has a unique global minimum in both the full and the reduced problem).

2.1.7 Kriging

An important and early application of Gaussian random field models has been termed *kriging* [112] after a South-African mining engineer D. Krige who developed methods for predicting spatial ore-grade distributions from sampled ore grades [93]. Optimal spatial linear prediction has its roots in earlier work by Wiener and Kolmogorov ("closeness in space" may have to be replaced by "closeness in time", since they were mainly concerned with time series). These fundamental ideas have been further developed in the fields of geostatistics [112] as kriging and in meteorology under the name *objective analysis* (see [36], Chap. 3 for references).

We will not go into any details, but refer to [36], Chap. 3 and [189] (we follow

the latter here). The basic model is the same as for semiparametric smoothing:

$$z(\boldsymbol{x}) = \boldsymbol{m}(\boldsymbol{x})^T\boldsymbol{\beta} + \varepsilon(\boldsymbol{x})$$

where $\boldsymbol{m}(\boldsymbol{x})$ is a known feature map and $\varepsilon(\boldsymbol{x})$ is a zero-mean random field with covariance function K. If $\varepsilon(\boldsymbol{x})$ is (weakly) stationary, $K(\boldsymbol{x})$ is also called *autocovariance function*. In a nutshell, kriging is a minimum mean-squared-error prediction method for linear functionals of $z(\boldsymbol{x})$ given observations $\boldsymbol{z} = (z(\boldsymbol{x}_1), \ldots, z(\boldsymbol{x}_n))^T$ at spatial locations $\boldsymbol{x}_i \in \mathbb{R}^d$. For example, if $z(\boldsymbol{x})$ measures ore grade at $\boldsymbol{x}$ one might be interested in predicting

$$\int_{\mathcal{B}} z(\boldsymbol{x})\, d\boldsymbol{x}$$

over some area $\mathcal{B} \subset \mathbb{R}^d$. Since they focus on m.s. error and m.s. properties of $z(\boldsymbol{x})$ in general, kriging methods typically depend on second-order properties of the process only, and $\varepsilon(\boldsymbol{x})$ is often assumed to be a Gaussian field. Furthermore, we restrict ourselves to linear predictors $\lambda_0 + \boldsymbol{\lambda}^T\boldsymbol{z}$. The optimal predictor of $z(\boldsymbol{x}_*)$ in the m.s. error sense is the conditional expectation which is linear in $\boldsymbol{z}$ if $\varepsilon(\boldsymbol{x})$ is Gaussian and $\boldsymbol{\beta}$ is known:

$$\boldsymbol{K}\boldsymbol{\lambda} = \boldsymbol{k}, \quad \lambda_0 = \left(\boldsymbol{m}(\boldsymbol{x}_*) - \boldsymbol{M}^T\boldsymbol{\lambda}\right)^T\boldsymbol{\beta}$$

where $\boldsymbol{K} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j}$, $\boldsymbol{k} = (K(\boldsymbol{x}_i, \boldsymbol{x}_*))_i$ and $\boldsymbol{M} = (\boldsymbol{m}(\boldsymbol{x}_1), \ldots, \boldsymbol{m}(\boldsymbol{x}_n))^T$. If $\boldsymbol{\beta}$ is unknown, a simple procedure is to plug in the generalised least squares estimate

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{M}^T\boldsymbol{K}^{-1}\boldsymbol{M}\right)^{-1}\boldsymbol{M}^T\boldsymbol{K}^{-1}\boldsymbol{z}$$

for $\hat{\boldsymbol{\beta}}$. This procedure can be motivated from several angles. If we restrict our attention to linear predictors of $z(\boldsymbol{x}_*)$ which are *unbiased* in the sense

$$\mathrm{E}\left[\lambda_0 + \boldsymbol{\lambda}^T\boldsymbol{z}\right] = \lambda_0 + \boldsymbol{\lambda}^T\boldsymbol{M}\boldsymbol{\beta} = \mathrm{E}[z(\boldsymbol{x}_*)] = \boldsymbol{m}(\boldsymbol{x}_*)^T\boldsymbol{\beta}$$

for any $\boldsymbol{\beta}$, the suggested approach minimises the m.s. error over these unbiased predictors. It is therefore called *best linear unbiased predictor (BLUP)*. A Bayesian motivation can be constructed in the same way as mentioned in Section 2.1.5. Namely, $\boldsymbol{\beta}$ is given a Gaussian prior whose covariance matrix scales

with $a > 0$ and $\varepsilon(\boldsymbol{x})$ is *a priori* Gaussian. Then, the posterior mean for $z(\boldsymbol{x}_*)$ converges to the BLUP as $a \to \infty$ (i.e. as the $\boldsymbol{\beta}$ prior becomes uninformative).

The equations behind the BLUP have been known long before and have been rediscovered in many areas of statistics. In practice, kriging methods are more concerned about inducing an appropriate covariance function (or autocovariance function under the stationarity assumption) from observed data as well. The empirical *semivariogram* is a frequently used method for estimating the autocovariance function close to the origin. On the theoretical side, Stein [189] advocates the usefulness of *fixed-domain asymptotics* to understand the relationship between covariance model and behaviour of kriging predictors.[29] By Bochner's theorem (2.2) an autocovariance function is characterised by its spectral distribution $F(\boldsymbol{\omega})$. Stein points out that fixed-domain asymptotics depend most strongly on the spectral masses for large $\|\boldsymbol{\omega}\|$, i.e. the high frequency components, much less so on the low frequency ones or the mean function $\boldsymbol{m}(\boldsymbol{x})^T\boldsymbol{\beta}$ (if $\boldsymbol{m}(\boldsymbol{x})$ is smooth itself, e.g. polynomials). Let $f(\boldsymbol{\omega})$ be the spectral density, i.e. the Fourier transform of $K(\boldsymbol{x})$. In general, the lighter the tails of $f(\boldsymbol{\omega})$ the smoother $\varepsilon(\boldsymbol{x})$ is in the m.s. sense. Stein advocates this expected smoothness as a central parameter of the GP prior and condemns the uncritical use of smooth (analytic) covariance functions such as the RBF (Gaussian) kernel (see Section 2.1.8). Another important concept highlighted by Stein (see also [200], Chap. 3) is the one of equivalence and orthogonality of GPs.[30] Essentially, GPs with autocovariance functions of different form can be equivalent in which case it is not possible to unambiguously decide for one of them even if an infinite amount of observations in a fixed region are given. On this basis, one can argue that for a parametric family of autocovariance functions inducing equivalent GPs the parameters can just as well be fixed *a priori* since their consistent estimation is not possible. On

[29] Stein restricts his analysis to "interpolation", i.e. to situations where predictions are required only at locations which are in principle supported by observations (in contrast to "extrapolation" often studied in the time series context). This should not be confused with the distinction between interpolation and smoothing used in Section 2.1.5. All non-trivial kriging techniques are smoothing methods.

[30] Two probability measure are equivalent if they have the same null sets, i.e. are mutually absolutely continuous (see Section A.1). They are orthogonal if there is a null set of one of them which has mass 1 under the other. Gaussian measures are either orthogonal or equivalent.

the other hand, parameters s.t. different values lead to orthogonal GPs should be learned from data and *not* be fixed *a priori*.

Note that kriging models are more generally concerned with *intrinsic random functions (IRF)* [113], generalisations of stationary processes which are also frequently used in the spline smoothing context. In a nutshell, a k-IRF $u(\boldsymbol{x})$ is a non-stationary random field based on a "spectral density" whose integral diverges on any environment of the origin (e.g., has infinite pointwise variance). However, if $\boldsymbol{c} \in \mathbb{R}^n$ is a generalised divided difference (g.d.d.) for $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in the sense that $\sum_i c_i p(\boldsymbol{x}_i) = 0$ for all polynomials p of total degree $\leq k$, then the variance of $\sum_i c_i u(\boldsymbol{x}_i)$ is finite and serves to define an "autocovariance function" $K(\boldsymbol{x})$ which is *conditionally positive semidefinite*, namely

$$\sum_{i,j=1}^{n} c_i \overline{c_j} K(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq 0$$

for all g.d.d.'s $\boldsymbol{c}$. In practice, one uses semi-parametric models where the latent process of interest is the sum of a k-IRF and a polynomial of total degree $\leq k$ whose coefficients are parametric latent variables. In fact, IRFs do not add more generality w.r.t. high-frequency behaviour of the process since $f(\boldsymbol{\omega})$ must be integrable on the complement of any $\mathbf{0}$-environment, so the IRF can be written as (uncorrelated) sum of a stationary and a non-stationary part, the latter with $f(\boldsymbol{\omega}) = 0$ outside a $\mathbf{0}$-environment (thus very smooth). IRFs are not discussed in any further detail in this thesis (see [113, 189]).

2.1.8 Choice of Kernel. Kernel Design

There is a tendency in the machine learning community to treat kernel methods as "black box" techniques, in the sense that covariance functions are chosen from a small set of candidates[31] over and over again. If a family of kernels is used, it typically comes with a very small number of free parameters so that model selection techniques such as cross-validation can be applied. Even though such approaches work surprisingly well for many problems of interest in machine learning, experience almost invariably has shown that much can be gained by choosing

[31] Some of which are not positive semidefinite, see below.

or designing covariance functions carefully depending on known characteristics of a problem (for an example, see [161], Sect. 11.4).

Establishing a clear link between kernel functions and consequences for predictions is very non-trivial and theoretical results are typically asymptotic arguments. As opposed to finite-dimensional parametric models, the process prior affects predictions from a non-parametric model even in fixed-domain asymptotic situations (see Section 2.1.7). The sole aim of this section is to introduce a range of frequently used kernel functions and some of their characteristics, to give some methods for constructing covariance functions from simpler elements, and to show some techniques which can be used to obtain insight into the behaviour of the corresponding GP. Yaglom [214] contains extensive material, an accessible review is [1]. In the final part, we discuss some kernel methods over discrete spaces $\mathcal{X}$.

It should be noted that positive definiteness of an arbitrary symmetric form or function is hard to establish in general. For example, the sensible approach of constructing a distance $d(\boldsymbol{x}, \boldsymbol{x}')$ between patterns depending on prior knowledge, then proposing

$$K(\boldsymbol{x}, \boldsymbol{x}') = e^{-w\, d(\boldsymbol{x}, \boldsymbol{x}')^2} \tag{2.26}$$

as covariance function does not work in general because K need not be positive semidefinite, moreover there is no simple criterion to prove or disprove K as covariance function.[32] If $d(\boldsymbol{x}, \boldsymbol{x}')$ can be represented in an Euclidean space, K is a kernel as we will see below. Note that if $K(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel, so must be $K(\boldsymbol{x}, \boldsymbol{x}')^t$ for any $t > 0$.[33] Kernels with this property are called *infinitely divisible*. Schönberg [162] managed to characterise infinitely divisible kernels (2.26) by a property on $d(\boldsymbol{x}, \boldsymbol{x}')$ which unfortunately is just as hard to handle as positive semidefiniteness.[34]

[32] If $d(\boldsymbol{x}, \boldsymbol{x}')$ is stationary, one can try to compute the spectral density, but this will not be analytically tractable in general.

[33] This is true in general only for $t \in \mathbb{N}_{>0}$, see below.

[34] $-d(\boldsymbol{x}, \boldsymbol{x}')^2$ must be conditionally positive semidefinite of degree 0 (see Section 2.1.7).

2.1.8.1 Constructing Kernels from Elementary Parts

We can construct complicated covariance functions from simple restricted ones which are easier to characterise (e.g. stationary or (an)isotropic covariance functions, see Section 2.1.1). A large number of families of elementary covariance functions are known (e.g., [214]), some of which are reviewed below.

A generalisation of stationary kernels to conditionally positive semidefinite ones (stationary fields to IRFs) is frequently used in geostatistical models (see Section 2.1.7) but will not be discussed here. The class of positive semidefinite forms has formidable closure properties. It is closed under positive linear combinations, pointwise product and pointwise limit. If $K(\boldsymbol{x}, \boldsymbol{x}')$ is a covariance function, so is

$$\tilde{K}(\boldsymbol{y}, \boldsymbol{y}') = \int h(\boldsymbol{y}; \boldsymbol{x}) h(\boldsymbol{y}'; \boldsymbol{x}') K(\boldsymbol{x}, \boldsymbol{x}') \, d\boldsymbol{x} d\boldsymbol{x}' \tag{2.27}$$

(if $\tilde{K}$ is finite everywhere). An important special case is $\tilde{K}(\boldsymbol{y}, \boldsymbol{y}') = a(\boldsymbol{x}) K(\boldsymbol{x}, \boldsymbol{x}')$ $a(\boldsymbol{x}')$. For example, a given kernel (with positive variance everywhere) can always be modified to be constant on the diagonal by choosing $a(\boldsymbol{x}) = K(\boldsymbol{x}, \boldsymbol{x})^{-1/2}$. Note that O'Hagan's "localized regression model" (Section 2.1.1) is also a special case of (2.27). A general way of creating a non-stationary covariance function $\tilde{K}(\boldsymbol{y}, \boldsymbol{y}')$ from a parametric model $h(\boldsymbol{y}; \boldsymbol{\theta})$ linear in $\boldsymbol{\theta}$ is to assume a GP prior on $\boldsymbol{\theta}$, then to integrate out the parameters (see [164] for details). Furthermore, suppose we do so with a sequence of models and priors to obtain a sequence of kernels. If the priors are appropriately scaled, the pointwise limit exists and is a kernel again. Many standard kernels can be obtained in this way (e.g., [207]). Neal showed that if the model size goes to infinity and the prior variances tend to 0 accordingly, layered models with non-Gaussian priors will also tend to a GP (due to the central limit theorem; see Section 2.1.1).

Another important modification is embedding. If $K(\boldsymbol{x}, \boldsymbol{x}')$ is a covariance function and $\boldsymbol{h}(\boldsymbol{y})$ is an arbitrary map, then

$$\tilde{K}(\boldsymbol{y}, \boldsymbol{y}') = K(\boldsymbol{h}(\boldsymbol{y}), \boldsymbol{h}(\boldsymbol{y}')) \tag{2.28}$$

is a covariance function as well (this is a special case of (2.27)). For example, if we have $d(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{h}(\boldsymbol{x}')\|$ in some Euclidean space, then (2.26)

is a valid kernel induced from the Gaussian (RBF) kernel (2.29). The Fisher kernel [81] and mutual information kernels [169] are examples. Embedding can be used to put rigid constraints on the GP. For example, if K is stationary in (2.28) and $\boldsymbol{h}(\boldsymbol{y}) = \boldsymbol{h}(\boldsymbol{y}')$, then $u(\boldsymbol{y}) = u(\boldsymbol{y}')$ almost surely.[35] For $\boldsymbol{h}(y) = (\cos((2\pi/\nu)y), \sin((2\pi/\nu)y))^T$, sample paths of $u(y)$ are ν-periodic functions.

2.1.8.2 Some Standard Kernels

In the following, we provide a list of frequently used "standard kernels". Most of these will have a *variance (scaling) parameter* $C > 0$ in practice, sometimes an *offset parameter* $v_b > 0$, thus instead of K one uses $C\,K$ or $C\,K + v_b$. C scales the variance of the process, while a $v_b > 0$ comes from the uncertainty of a bias parameter added to the process.[36] In applications where the kernel matrix $\boldsymbol{K}$ is used directly in linear systems, it is advised to add a *jitter term*[37] $\rho\delta_{\boldsymbol{x},\boldsymbol{x}'}$ to the kernel to improve the condition number of $\boldsymbol{K}$. This amounts to a small amount of additive white noise on $u(\boldsymbol{x})$ (ρ can be chosen quite small), but should not be confused with measurement noise which is modelled separately (see Section 2.1.2).[38] These modifications are omitted in the sequel for simplicity.

The *Gaussian (RBF)* covariance function

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{w}{2d}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right) \tag{2.29}$$

is isotropic for each $\mathcal{X} = \mathbb{R}^d$ (i.e. $\mathcal{D}^\infty$). $w > 0$ is an inverse squared length scale parameter, in the sense that $w^{-1/2}$ determines a scale on which $u(\boldsymbol{x})$ is expected to change significantly. $K(\boldsymbol{x})$ is analytic at $\mathbf{0}$, so $u(\boldsymbol{x})$ is m.s. analytic. Stein [189] points out that

$$\sum_{j=0}^{k} u^{(j)}(0)\frac{x^j}{j!} \to u(x)$$

[35] This is because the correlation $\rho(u(\boldsymbol{y}), u(\boldsymbol{y}'))$ is 1, thus $u(\boldsymbol{y}) = a\,u(\boldsymbol{y}') + b$ for fixed a, b, then $a = 1, b = 0$ because both variables have the same mean and variance.

[36] For reasons of numerical stability, v_b must not become too large.

[37] In the context of kriging (see Section 2.1.7), adding $\rho\delta_{\boldsymbol{x},\boldsymbol{x}'}$ has been proposed by Mathéron to model the so-called "nugget effect" (see [36], Sect. 2.3.1), but other authors have criticised this practice.

[38] The probit noise model (2.9) is obtained by adding $N(0,1)$ noise and passing the result through a Heaviside step.

in quadratic mean for every x (a similar formula holds for $\mathcal{X} = \mathbb{R}^d$), so that u can be predicted perfectly by knowing all its derivatives at $\mathbf{0}$ (which depend on u on an environment of $\mathbf{0}$ only). He criticises the wide-spread use of the Gaussian covariance function because its strong smoothness assumptions are unrealistic for many physical processes, in particular predictive variances are often unreasonably small given data. The spectral density (in $\mathbb{R}$) is $f(\omega) = (2\pi w)^{-1/2} \exp(-\omega^2/(2w))$ with very light tails. On the other hand, Smola et. al. [182] recommend the use of the Gaussian covariance function for high-dimensional kernel classification methods because of the high degree of smoothness.[39] It is interesting to note that in the context of using GPs for time series prediction, Girard et. al. [65] report problems with unreasonably small predictive variances using the Gaussian covariance function (although they do not consider other kernels in comparison). Figure 2.1 shows smoothed plots of some sample paths. Note the effect of the length scale $w^{-1/2}$ and the high degree of smoothness.

We can consider the anisotropic version, called *squared-exponential* covariance function:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2d}(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{x}')\right). \tag{2.30}$$

Here, $\boldsymbol{W}$ is positive definite. Typically, $\boldsymbol{W}$ is a diagonal matrix with an inverse squared length scale parameter w_j for each dimension. Full matrices $\boldsymbol{W}$ have been considered in [144, 198], and factor analysis-type matrices $\boldsymbol{W}$ are a useful intermediate (e.g., [11, 168]). An important application of the additional d.o.f.'s in (2.30) as compared to the Gaussian kernel is automatic relevance determination (ARD), as discussed below. Note that the squared-exponential covariance function can be seen as product of d one-dimensional Gaussian kernels with different length scales, so the corresponding RKHS is a tensor product space built from RKHS's for one-dimensional functions (see Section 2.1.4).

The *Matérn class* of covariance functions (also called *modified Bessel* covariance functions) is given by

$$K(\tau) = \frac{\pi^{1/2}}{2^{\nu-1}\Gamma(\nu + 1/2)\alpha^{2\nu}} (\alpha\tau)^\nu K_\nu(\alpha\tau), \quad \tau = \|\boldsymbol{x} - \boldsymbol{x}'\|, \tag{2.31}$$

[39] Most experiments in this thesis use the Gaussian covariance function.

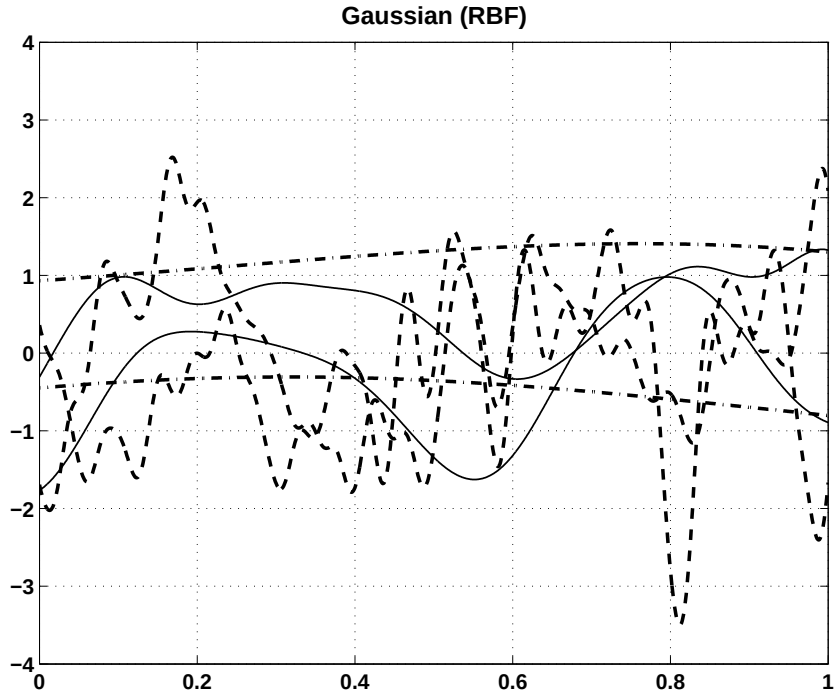

Figure 2.1: Smoothed sample paths from GP with Gaussian covariance function. All have variance $C = 1$. Dash-dotted: $w = 1$. Solid: $w = 10^2$. Dashed: $w = 50^2$.

where $\nu > 0$, $\alpha > 0$ and $K_\nu(x)$ is a modified Bessel function (e.g., [189], Sect. 2.7). One can show that $z^\nu K_\nu(z) \to 2^{\nu-1}\Gamma(\nu)$ for $z \to 0$, so

$$K(0) = \frac{\pi^{1/2}\Gamma(\nu)}{\Gamma(\nu + 1/2)\alpha^{2\nu}}.$$

K is isotropic for each $\mathcal{X} = \mathbb{R}^d$. An important feature of this class is that the m.s. smoothness of $u(\boldsymbol{x})$ can be regulated directly via ν. For example, $u(\boldsymbol{x})$ is m times m.s. differentiable iff $\nu > m$. The spectral density in $\mathbb{R}$ is $f(\omega) = (\alpha^2 + \omega^2)^{-\nu-1/2}$. For $\nu = 1/2 + m$ we obtain a process with rational spectral density, a continuous time analogue of an ARMA time series model. For $\nu = 1/2$, $K(\tau) \propto e^{-\alpha\tau}$ defines an *Ornstein-Uhlenbeck process*, a stationary analogue to the Wiener process which also has independent increments. In general, for $\nu = 1/2 + m$ we have $K(\tau) \propto e^{-\alpha\tau}p(\alpha\tau)$, where $p(x)$ is a polynomial of order m with constant coefficient 1 (e.g., [189], Sect. 2.7). Note that if $\alpha = (w(2\nu+1))^{1/2}$,

then

$$\alpha^{2\nu+1} f(\omega) \to e^{-\omega^2/(2w)} \ (\nu \to \infty),$$

thus $K(\tau)$ converges to the Gaussian covariance function after appropriate re-scaling.

The Matérn class can be generalised to an anisotropic family in the same way as the Gaussian kernel. Figure 2.2 show some sample function plots for values $\nu = 1/2, 3/2, 5/2, 10$. Note the effect of ν on the roughness of the sample paths. For $\nu = 1/2$ the paths are erratic even though the length scale is 1, i.e. the same as the horizontal region shown. For $\nu = 3/2$, the process is m.s. differentiable, for $\nu = 5/2$ twice so.

The *exponential class* of covariance functions is given by

$$K(\tau) = e^{-\alpha\tau^\delta}, \ \delta \in (0, 2].$$

The positive definiteness can be proved using the Matérn class (see [189], Sect. 2.7). For $\delta = 1$, we have the Ornstein-Uhlenbeck covariance function, for $\delta = 2$ the Gaussian one. Although it seems that the kernel varies smoothly in δ, the processes have quite different properties in the regimes $\delta \in (0, 1)$, $\delta = 1$, $\delta \in (1, 2)$ and $\delta = 2$. Continuous sample paths can be ensured for any $\delta \in (0, 2]$, but differentiable sample paths can only be obtained for $\delta = 2$ (in which case they are analytic).[40] $K(\tau)$ is not positive definite for $\delta > 2$. Figure 2.3 shows some sample path plots.

We have derived the *spline* covariance function on $[0, 1]$ (2.22) from first principles above. This kernel is of interest because posterior mean functions in GP models (or minimisers of the variational problem over the RKHS) are splines of order m, i.e. piecewise polynomials in C^{2m-2} (see Section 2.1.5) and associated computations are $O(n)$ (where n is the number of training points, or "knots") only. On the other hand, technical complications arise because spline kernels are RKs for subspaces of $\mathcal{W}_m[0, 1]$ only, namely of the functions which satisfy the boundary conditions (see Section 2.1.5). The operator induced by a spline kernel has a null space spanned by polynomials, and in practice it is necessary to adjoin

[40] All these statements hold with probability 1, as usual.

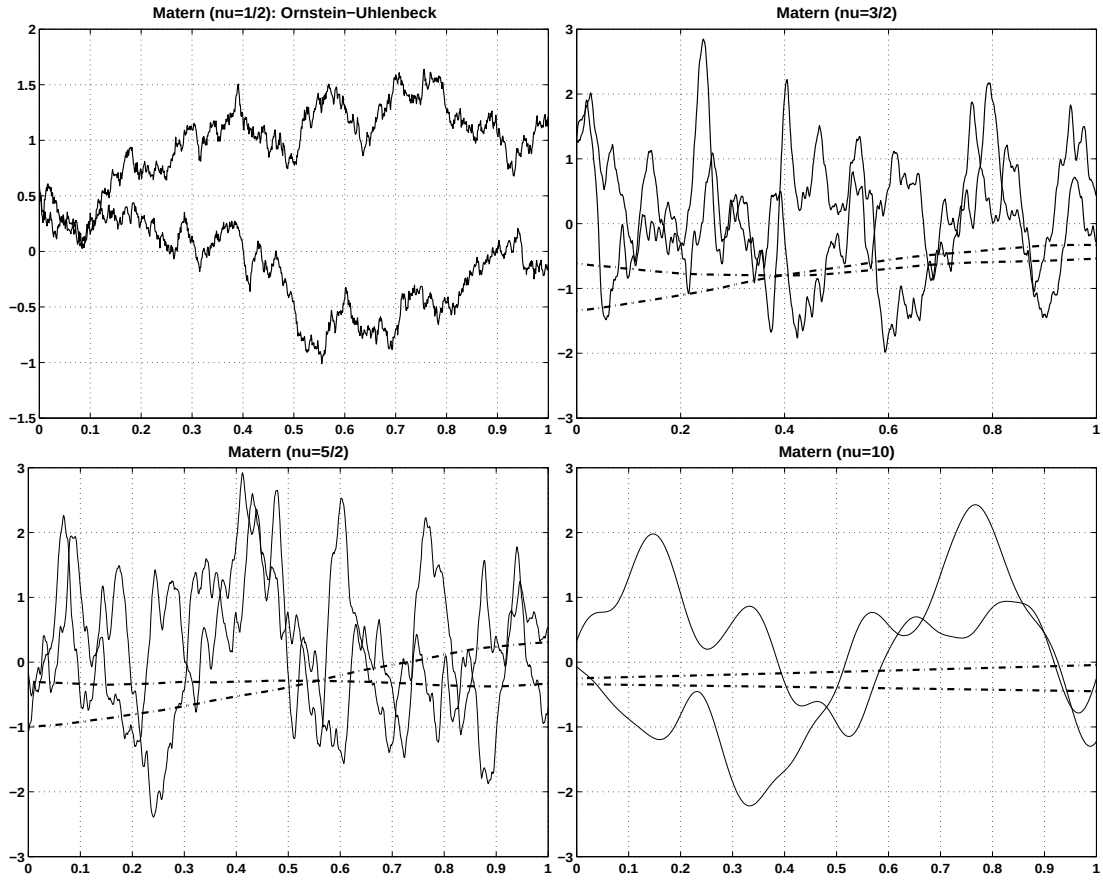

Figure 2.2: Smoothed sample paths from GP with Matérn covariance function. All have variance $C = 1$. Upper left: Ornstein-Uhlenbeck (Matérn, $\nu = 1/2$), $w = 1$. Upper right: Matérn, $\nu = 3/2$, $w = 1$ (dash-dotted), $w = 10^2$ (solid). Lower left: Matérn, $\nu = 3/2$, $w = 1$ (dash-dotted), $w = 10^2$ (solid). Lower right: Matérn, $\nu = 10$, $w = 1$ (dash-dotted), $w = 10^2$ (solid).

the corresponding (finite-dimensional) space. The spline kernels are not stationary (they are supported on $[0, 1]$), but we can obtain spline kernels on the circle by imposing periodic boundary conditions on $\mathcal{W}_m[0, 1]$, leading to the stationary kernel

$$K(x, x') = \sum_{\nu \geq 1} \frac{2}{(2\pi\nu)^{2m}} \cos(2\pi\nu(x - x')).$$

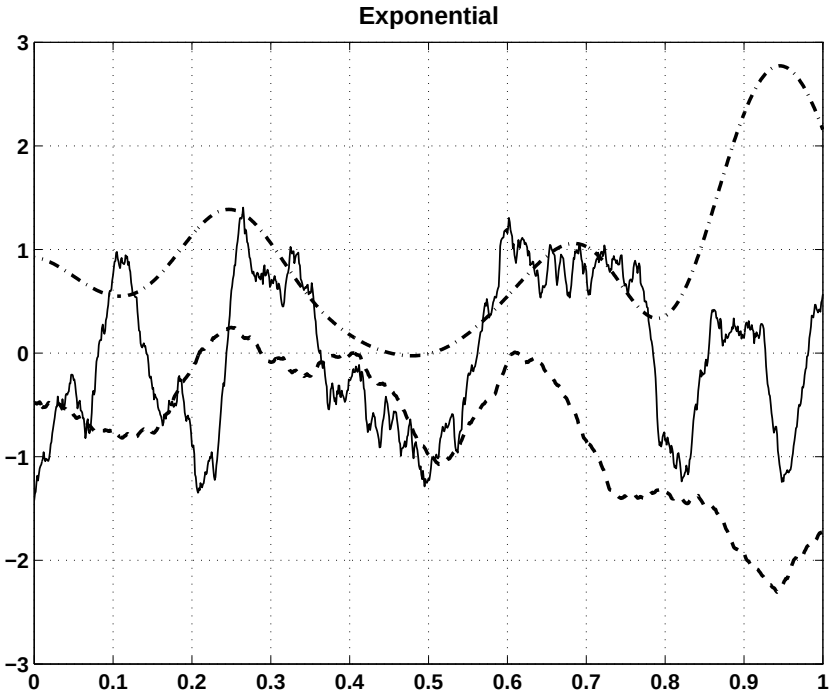

Figure 2.3: Smoothed sample paths from GP with exponential covariance function. All have variance $C = 1$ and $\alpha = 10^2$. Solid: $\delta = 1.5$. Dashed: $\delta = 1.9$. Dash-dotted: $\delta = 2$ (Gaussian).

From this representation, it follows that the spectral density is

$$f(\omega) = \sum_{\nu \geq 1} \frac{1}{(2\pi\nu)^{2m}} \delta_{2\pi\nu}(|\omega|)$$

which is discrete. Note that sample functions from $u(x)$ are periodic with probability 1. In Wahba [200], Chap. 2 it is shown how to construct splines on the sphere by using the iterated Laplacian, but this becomes quite involved. An equivalent to splines (in a sense) can be defined in $\mathbb{R}^d$ using *thin-plate spline* conditionally positive definite functions (see Section 2.1.7), see [200, 66] for details.

For kernel discrimination methods, *polynomial* covariance functions

$$K(\boldsymbol{x}, \boldsymbol{x}') = \frac{(\boldsymbol{x}^T \boldsymbol{x}' + \alpha)^m}{((\|\boldsymbol{x}\|^2 + \alpha)(\|\boldsymbol{x}'\|^2 + \alpha))^{m/2}}, \quad \alpha \geq 0, m \in \mathbb{N}$$

are popular although they seem unsuitable for regression problems. Polynomial kernels can be seen to induce a finite-dimensional feature space of polynomials

of total degree $\leq m$ (if $\alpha > 0$). It is interesting to note that this is exactly the RKHS we have to adjoin to one for a conditionally positive definite kernel of order m such as the thin-plate spline covariance function. On the other hand, in the spline case these polynomial parts are usually not regularised at all. By the Karhunen-Loeve expansion (see Section 2.1.4), we can write $u(\boldsymbol{x})$ as expansion in all monomials of total degree $\leq m$ with Gaussian random coefficients. The regularisation operator (see Section 2.1.5) for polynomial kernels is worked out in [182]. Note that $K(\boldsymbol{x}, \boldsymbol{x}')$ is not a covariance function for $m \notin \mathbb{N}$, thus the kernel is not infinitely divisible. Figure 2.4 shows some sample path plots. These are polynomials and therefore analytic.

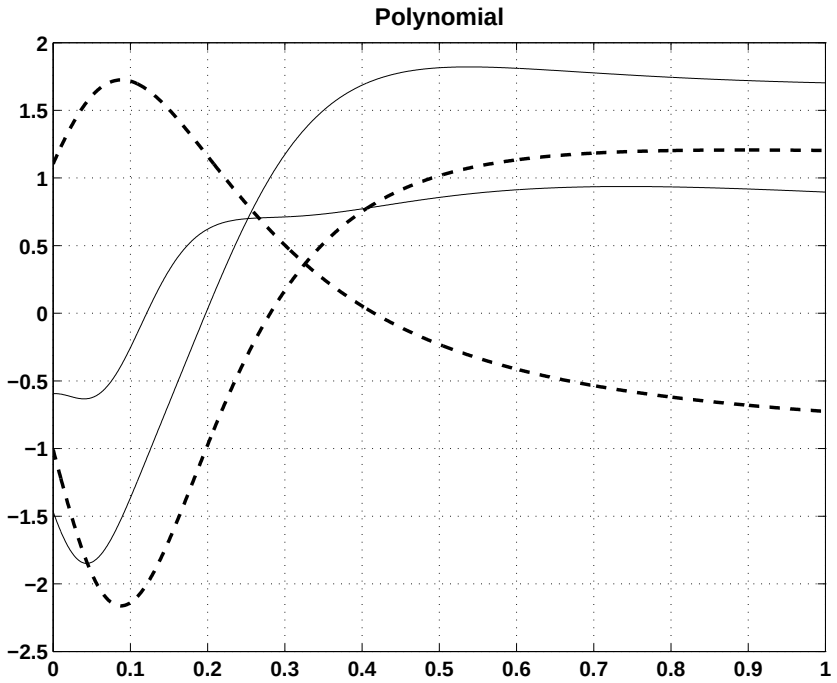

Figure 2.4: Sample paths from GP with polynomial covariance function. All have variance $C = 1$ and $\alpha = 0.05$. Solid: $m = 10$. Dashed: $m = 5$.

The Euclidean inner product $\boldsymbol{x}^T \boldsymbol{\Sigma} \boldsymbol{x}'$ is sometimes referred to as "linear kernel" in the machine learning literature. GP models based on this kernel are nothing else than straightforward linear models (linear regression, logistic regression, etc.). It is clear from the weight space view (see Section 2.1.1) that a linear model can

always be regarded as a GP model (or kernel technique), but this makes sense only if $n < d$, where n is the number of training points.[41] Furthermore, the SVM with linear kernel is a variant of the perceptron method [154] with "maximal stability" [156] studied in statistical physics.

Finally, let us give an example of a function which is *not* a covariance function, the so-called "sigmoid kernel"

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tanh\left(a\boldsymbol{x}^T\boldsymbol{x}' + b\right).$$

K is not positive semidefinite for any a, b (see [180]), it is nevertheless shipped in most SVM packages we know of. It springs from the desire to make kernel expansions look like restricted one-layer neural networks. The correct link between MLPs and GP models has been given by Neal (see Section 2.1.1), which involves taking the limit of infinitely large networks. A covariance function corresponding to a one-layer MLP in the limit has been given by Williams [207]. In practice, it is of course possible to fit expansions of kernels to data which are not covariance functions. However, the whole underlying theory of minimisation in a RKHS (see Sections 2.1.4 and 2.1.5) breaks down, as does the view as inference in a GP model. On the practical side, flawed results such as negative predictive variances can pop up when least expected. Even worse, most optimisation techniques (including SVM algorithms) rely on the positive semidefiniteness of matrices and may break down otherwise.

2.1.8.3 Guidelines for Kernel Choice

Choosing a good kernel for a task depends on intuition and experience. On high-dimensional tasks where no suitable prior knowledge is available, the best option may be to explore simple combinations of the standard kernels listed above. If invariances are known, they may be encoded using the methods described in [161], Sect. 11.4. With approximate Bayesian GP inference methods as discussed in this thesis, one can in principle use combinations of different kernels with a lot of free (hyper)parameters which can be adapted automatically.

[41] Otherwise, running a kernel algorithm is wasteful and awkward due to a singular kernel matrix.

For low-dimensional $\mathcal{X}$, one can obtain further insight. Stein [189] points out the usefulness of studying fixed-domain asymptotics (see Section 2.1.7). In this respect, the tail behaviour of the spectral density (see Section 2.1.1) is important. The m.s. degree of differentiability (degree of smoothness) of the process depends on the rate of decay of $f(\boldsymbol{\omega})$. Stein recommends kernel families such as the Matérn class (2.31) which come with a degree of smoothness parameter ν. He also stresses the importance of the concept of equivalence and orthogonality of GPs (see Section 2.1.7), arguing that if a set of kernels have equivalent processes, one might as well fix any of them *a priori* because even with an infinite amount of data we cannot separate between them with absolute certainty. Both of these arguments are asymptotic. For example, it is not clear whether ν in the Matérn class can be learned accurately enough from a limited amount of data. Also, predictions from equivalent processes with different kernels can be different.[42]

There are ways of "getting a feeling" for the behaviour of a process by visualisation, which is an option if $\mathcal{X} = \mathbb{R}^d$ is low-dimensional, $d = 1, 2$. We can draw "samples" from the process and plot them as follows (the plots in this section have been produced in this way). Let $X \subset \mathcal{X}$ be a fine grid[43] over a domain of interest, $n = |X|$ and $\boldsymbol{u} = u(X) \sim N(\boldsymbol{0}, \boldsymbol{K}(X))$. We can sample $\boldsymbol{u}$ as $\boldsymbol{u} = \boldsymbol{L}\boldsymbol{v},\ \boldsymbol{v} \sim N(0, \boldsymbol{I})$, where $\boldsymbol{K}(X) = \boldsymbol{L}\boldsymbol{L}^T$ is the Cholesky decomposition (Definition A.2). If X is too large, $\boldsymbol{u}$ can be approximated using an incomplete Cholesky factorisation of $\boldsymbol{K}(X)$ (see [212]). If $d = 1$, the process is isotropic and the grid is regularly spaced, $\boldsymbol{K}(X)$ has *Toeplitz* structure[44] and its Cholesky decomposition can be computed in $O(n^2)$ (see [44]). Repeatedly sampling $\boldsymbol{u}$ and plotting $(X, \boldsymbol{u})$ can give an idea about degree of smoothness, average length scales (Euclidean distance in $\mathcal{X}$ over which $u(\boldsymbol{x})$ is expected to vary significantly) or other special features of K.

[42] Stein argues (citing Jeffreys) that such differences cannot be important since they do not lead to consistency in the large data limit (in a fixed domain).

[43] For fine grids and smooth kernels such as the Gaussian one, the Cholesky technique described here fails due to round-off errors. The singular value decomposition (SVD) should be used in this case, concentrating on the leading eigendirections which can be determined reliably.

[44] A matrix is Toeplitz if all its diagonals (main and off-diagonals) are constant.

2.1.8.4 Learning the Kernel

One promising approach for choosing a covariance function is to learn it from data and/or prior knowledge. For example, given a parametric family of covariance functions, how can we choose[45] the parameters in order for the corresponding process to model the observed data well?

Model selection from a fixed family can be done by the empirical Bayesian method of marginal likelihood maximisation (see Section A.6.2), a generic approximation of which in the case of GP models is given in Section 2.1.3. Since this procedure typically scales linearly in the number of hyperparameters, elaborate and heavily parameterised families can be employed. An important special case has been termed *automatic relevance determination (ARD)* by MacKay [109] and Neal [131]. The idea is to introduce a hyperparameter which determines the scale of variability of a related variable of interesting (with prior mean 0). For example, we might set up a linear model (2.4) by throwing in a host of different features (components in $\Phi(\boldsymbol{x})$), then place a $N(\boldsymbol{\beta}|\mathbf{0}, \boldsymbol{D})$ on the weights $\boldsymbol{\beta}$ where $\boldsymbol{D}$ is a diagonal matrix of positive hyperparameters. If we place a hyperprior on $\operatorname{diag} \boldsymbol{D}$ which encourages small values, there is an *a priori* incentive for $d_i = \boldsymbol{D}_{i,i}$ to become very small, inducing a variance of β_i close to 0 which effectively switches off the effect of $\beta_i \phi_i(\boldsymbol{x})$ on predictions. This is balanced against the need to use at least some of the components of the model to fit the data well, leading to an automatic discrimination between relevant and irrelevant components. In the context of covariance functions, we can implement ARD with any anisotropic kernel (see Section 2.1.1) of the form

$$K(\boldsymbol{x}, \boldsymbol{x}') = \tilde{K}((\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{W} (\boldsymbol{x} - \boldsymbol{x}')),$$

where $\tilde{K}$ is isotropic and $\boldsymbol{W}$ is diagonal and positive definite. An example is the squared-exponential covariance function (2.30). Here, w_i determines the scale of variability of the (prior) field as $\boldsymbol{x}$ moves along the i-th coordinate axis. If we imagine the field being restricted to a line parallel to this axis, $w_i^{-1/2}$ is the

[45] The proper Bayesian solution would be to integrate out the parameters (see Section A.6.2), but even if this can be approximated with MCMC techniques, the outcome is a mixture of covariance functions leading to expensive predictors.

length scale of this restriction, i.e. a distance for which the expected change of the process is significant. If $w_i \approx 0$, this length scale is very large, thus the field will be almost constant along this direction (in regions of interest). Thus, via ARD we can discriminate relevant from irrelevant dimensions in the input variable $\boldsymbol{x}$ automatically, and predictions will not be influenced significantly by the latter. Section 4.8.2 provides an example for how important ARD can be in the context of difficult real-world GP regression problems with high-dimensional input spaces.

In spatial statistics, semivariogram techniques (see [36], Sect. 2.3.1) are frequently used. For a stationary process, the (semi)variogram is $\gamma(\boldsymbol{x} - \boldsymbol{x}') = (1/2)\mathrm{Var}[u(\boldsymbol{x}) - u(\boldsymbol{x}')]$. It is estimated by averaged squared distances over groups of datapoints which are roughly the same distance apart and fitted to parametric families by maximum likelihood. Stein [189] criticises the use of the empirical semivariogram as single input for choosing a covariance function, which is actually impossible if the field is m.s. differentiable and gives a range of other techniques, including the empirical Bayesian approach mentioned above.

For classification models, the idea of local invariance w.r.t. certain groups of transformations is important. For example, the recognition of handwritten digits should not be influenced by translations or small-angle rotations of the bitmap.[46] If a process is used as latent function in a classification problem, e.g. representing the log probability ratio between classes (see Section 2.1.2), then starting from some $\boldsymbol{x}$ and applying small transformations from a group w.r.t. which discrimination should remain invariant should not lead to significant changes in the process output (e.g. in the m.s. sense). To relate this notion to ARD above, varying $\boldsymbol{x}$ along such invariant directions should induce a coordinate of $\boldsymbol{x}$ (non-linear in general) which is irrelevant for prediction. Chapter 11 in [161] gives a number of methods for modifying a covariance function in order to incorporate invariance knowledge to some degree, also reviewing work in that direction which we omit here.

Finally, Minka [122] pointed out that instances of the "learning how to learn"

[46] Although a 180-degree rotation of a 6 results in a 9.

or "prior learning" paradigm can be seen as learning a GP prior from multi-task data (see his paper for references). In fact, the setup is the one of a standard hierarchical model frequently used in Bayesian statistics to implement realistic prior distributions. We have access to *several* noisy samples and make the assumption that these have been sampled from different realisations of the latent process which in turn have been sampled i.i.d. from the process prior. Data of this sort is very valuable for inferring aspects of the underlying covariance function. In a simple multi-task scenario a multi-layer perceptron is fit to several samples by penalised maximum likelihood, sharing the same input-to-hidden weights but using different sets of hidden-to-output weights for each sample. The idea is that the hidden units might discover features which are important in general, while the combination in the uppermost layer is specific. If we place Gaussian priors on the hidden-to-output weights, this becomes a GP model with a covariance function determined by the hidden units. More generally, we can start from any parametric family of covariance functions and learn hyperparameters or even the hyperposterior from multi-task data using marginal likelihood maximisation together with the hierarchical sampling model. An approximate implementation of this idea has been reported in [141].

2.1.8.5 Kernels for Discrete Objects

As mentioned in Section 2.1.1, in principle the input space $\mathcal{X}$ is not restricted to be $\mathbb{R}^d$ or even a group. For example, Gaussian processes over lattices are important in vision applications (in the form of a Gaussian Markov random field with sparse structured inverse covariance matrix). For Gaussian likelihoods, the posterior mean can be determined most efficiently using a conjugate gradients solver[47] and the embedded trees algorithm of Wainwright, Sudderth and Willsky [202] can be used to compute the marginal variances as well. Kernel methods, i.e. methods which use covariance matrices over variables determined from the "spatial" relationship of these (or associated covariates) have been proposed for

[47] Loopy belief propagation renders the correct mean as well if it converges [206], but is much slower and often numerically unstable.

a number of problems involving discrete spaces $\mathcal{X}$ (finite or countably infinite). Our aim in this section is no more than to give a few selected examples.

Kernels can be defined on the set of finite-length strings from a finite alphabet. Many *string kernels* have been proposed recently, but we will not try to review any of this work. Important applications of string kernels (or distance measures between sequences) arise from problems in DNA or RNA biology where statistical models have to be build for nucleotide sequences. Many proposed string kernels are special cases of *convolution kernels* introduced by Haussler [69]. Maybe the most interesting case discussed there is the extension of a hidden Markov random field (HMRF). The latter is a Markov random field (MRF) with observed variables $\boldsymbol{x}$, latent variables $\boldsymbol{u}$ and clique potentials $C_d(x_d, u_d)$ where x_d, u_d are subsets of components of $\boldsymbol{x}$, $\boldsymbol{u}$, and $\boldsymbol{u}$ is marginalised over. If we replace the clique potential by positive definite kernels $K_d((x_d, u_d), (y_d, v_d))$ and marginalise over $\boldsymbol{u}$, $\boldsymbol{v}$, the result is a covariance kernel which can also be seen as unnormalised joint generative distribution for $(\boldsymbol{x}, \boldsymbol{y})$. If the original MRF has a structure which allows for tractable computation, the same algorithm can be used to evaluate the covariance function efficiently. For example, a hidden Markov model (HMM) for sequences can be extended to a pair-HMM in this way, emitting two observed sequences sharing the same latent sequence, and many string kernels arise as special cases of this construction.

In practice, string kernels (and more generally kernels obtained from joint probabilities under pair-HMRFs) often suffer from the "ridge problem": $K(\boldsymbol{x}, \boldsymbol{x})$ is much larger than $K(\boldsymbol{x}, \boldsymbol{x}')$ for many $\boldsymbol{x}'$ for which *a priori* we would like to attain a significant correlation, especially if rather long sequences are compared. For example, in models involving DNA sequences we would like sequences to correlate strongly if they are homologuous, *i.e.* encode for proteins of very similar function. In a standard string kernel, two sequences are strongly correlated if both can be obtained from a common "ancestor" latent sequence by few operations such as insertions and substitutions (this ancestor model is motivated by the evolution of genes and gives a good example for the pair-HMM setup). However, often homologuous sequences differ quite substantially in regions on which the structure

of the functional part of the protein does not depend strongly. Such "remote" homologies are the really interesting ones, since very close homologies can often be detected using simpler statistical techniques than process models based on string kernels. On the other hand, it may be possible to spot such homologies by going beyond string kernels and pair-HMRF constructions, for example building on the general framework given in [32] where kernels are obtained from finite transducers.

A conceptually simple way to obtain a kernel on $\mathcal{X}$ is to embed $\mathcal{X}$ in some Euclidean space $\mathbb{R}^d$, then to concatenate the embedding with any of the known $\mathbb{R}^d$ kernels, for example the Gaussian one (2.29). An example is the Fisher kernel [81] which maps datapoints to their "Fisher scores" under a parametric model. There has been a surge of interest recently in automatic methods for parameterising low-dimensional non-linear manifolds (e.g., [191, 155]) by local Euclidean coordinates. Although these methods are non-parametric, they can be used to fit conventional parametric mixture models in order to obtain a parametric embedding which could then be used to obtain a kernel.

Recently, Kondor and Lafferty [92] proposed kernels on discrete objects using concepts from spectral graph theory (diffusion on graphs). If $\mathcal{X}$ is finite, a covariance function on $\mathcal{X}$ is simply a positive semidefinite matrix. If $\boldsymbol{H}$ is a symmetric *generator matrix*, the corresponding *exponential kernel* is defined as

$$\boldsymbol{K}^{(\beta)} = \exp(\beta \boldsymbol{H}) = \sum_{j \geq 1} \frac{\beta^j}{j!} \boldsymbol{H}^j, \quad \beta \geq 0. \tag{2.32}$$

We define $K_\beta(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{K}^{(\beta)}_{\boldsymbol{x},\boldsymbol{x}'}$, where we use elements of $\mathcal{X}$ as indices into the matrix $\boldsymbol{K}^{(\beta)}$. $\boldsymbol{K}^{(\beta)}$ is positive definite. In fact, it has the same eigenvectors as $\boldsymbol{H}$, but the eigenspectrum is transformed via $\lambda \to \exp(\beta\lambda)$. In practice, general exponential kernels cannot be computed feasibly if $\mathcal{X}$ is large, especially there is no general efficient way of computing kernel matrices of K_β over points of interest. It might be possible to approximate marginalisations of $\boldsymbol{K}^{(\beta)}$ by sampling. The kernel and generator matrices are linked by the *heat equation*

$$\frac{\partial}{\partial \beta} \boldsymbol{K}^{(\beta)} = \boldsymbol{H}\, \boldsymbol{K}^{(\beta)}.$$

It is interesting to note that every infinitely divisible covariance function on $\mathcal{X}$ has the form (2.32), namely if $K_\beta = K_1^\beta$ with covariance matrix $\boldsymbol{K}$ then $\boldsymbol{H} = \partial \boldsymbol{K} / \partial \beta$ at $\beta = 0$. Kondor and Lafferty are interested in *diffusion kernels* on graphs as special cases of exponential kernels. Here, the generator is the negative of the so-called graph Laplacian. The construction can be seen as stationary Markov chain (random walk) in continuous time on the vertices of the graph. The kernel $K_\beta(\boldsymbol{x}, \boldsymbol{x}')$ is the probability of being at $\boldsymbol{x}'$ at time β, given that the state at time 0 was $\boldsymbol{x}$. This interpretation requires that $\boldsymbol{H}\mathbf{1} = \mathbf{0}$ which is true for the negative graph Laplacian and which implies that $\boldsymbol{K}^{(\beta)}$ is (doubly) stochastic. The same equation describes heat flow or diffusion from an initial distribution. The idea is to describe the structure of $\mathcal{X}$ (in the sense of "closeness", *i.e.* close points should be highly correlated under the covariance function) in terms of local neighbourhood association which induce an (weighted or unweighted) undirected graph. Then, the correlation at some $\boldsymbol{x}$ with all other points is proportional to the distribution of a random walk started at $\boldsymbol{x}$ after time β. Similar ideas have been used very effectively for non-parametric clustering. Kondor and Lafferty give examples for graph of special regular structures for which the diffusion kernel can be determined efficiently. These include certain special cases of string kernels (here, $\mathcal{X}$ is infinite and the analogue to Markov chains has to be treated more carefully). In situations where K_β cannot be determined by known simple recursive formulae, one could represent $\mathcal{X}$ by a representative sample including the training set (but also unlabelled data). If the generator matrix of the underlying graph (projected onto the representative sample in a sensible way) is sparse, its leading eigenvectors and eigenvalues could be approximated by sparse eigensolvers which would lead to an approximation of $\boldsymbol{K}^{(\beta)}$ which is low-rank optimal w.r.t. the Frobenius norm. Kondor and Lafferty also note that on the graph given by a regular grid in $\mathbb{R}^d$, the generator matrix converges towards the usual Laplacian operator and K_β towards the Gaussian kernel (2.29) as the mesh size approaches 0.

2.2 Learning Theory

The field of *computational learning theory (COLT)* is concerned with two basic questions. Given a particular problem setup, described typically as a set of related random variables, the goal is to predict future outcomes of a particular variable of interest. The *accuracy* of a prediction can be quantified in a frequentist way by imagining that the variable and its prediction are sampled repeatedly and independently and counting the number of times the prediction is correct.[48] Suppose we observe an i.i.d. sample[49] of the problem at hand. *How many sample points* do we need to learn a predictor which attains a desired accuracy? *How much computation* do we have to spend to learn such a predictor? The first question asks for the *sample complexity* of the problem. The second question, although of major importance in practice, is not discussed here in any further detail. The subfield of COLT which ignores running time complexity issues is sometimes called *statistical learning theory (SLT).*

This thesis describes the application of a learning-theoretical result to nonparametric classification (see Chapter 3), but it turns out that in order to understand the general result and its specialisation to the case of our interest, most of the mathematical concepts used in learning theory today are not required at all. Therefore, this introduction to some aspects of learning theory aims for no more than giving a flavour of the inner workings of PAC and VC results. Readers familiar with these topics need not spend any time here (they may want to glance at Section 2.2.1 where our notation is defined). There are many sources the interested reader can tap for thorough introductions to learning theory, *e.g.* [88, 197, 4, 3]. An excellent book about distribution-free analyses of learning algorithms is Devroye et. al. [49] which we make use of here, see also [68]. The annual COLT conference is a way to get familiar with recent work. It will become clear that many aspects of SLT are implicit in the wider mathematical field of *empirical process theory.* For the reader interested in a specific treatment

[48] If the target variable is from an infinite set, we can specify a loss function and a threshold in advance and define the prediction to be correct iff the loss falls below the threshold.

[49] The framework can be extended to non-i.i.d. sampling with a known structure, e.g. Markovian, but this is not discussed here.

for kernel algorithms such as SVM, a number of books is available [161, 72, 37].

The structure of this section is as follows. In Section 2.2.1, we introduce the PAC framework in the case we are interested in. Section 2.2.2 contains a short account of concentration inequalities. In Section 2.2.3, we give a brief account of the notions of VC theory for binary classification. Finally, in Section 2.2.4 we comment on the applicability of PAC results for statistical model selection.

Note that our exposition in parts follows closely the excellent tutorial on learning theory for classification of Gábor Lugosi [105].

2.2.1 Probably Approximately Correct

In this section, we set out to formally define the framework many COLT problems are studied in. We restrict ourselves to the binary classification problem introduced in Section A.5, where the training sample is $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\}$, $\boldsymbol{x}_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$. We also concentrate on the problem of finding PAC results for a *specific* learning method of interest. Trivially, such a result gives an upper bound on the sample complexity for a problem. Corresponding lower bounds can be found typically as well, but this is not in the scope of this work.

PAC analyses are *distribution-free*: we assume that S is generated i.i.d. from some unknown *data distribution*, but a PAC analysis is not conditioned on any assumptions about this distribution, its result must hold for *any* such source. The variable we are most interested in is the *generalisation error* $\mathrm{gen}(S)$ (see Section A.5), *i.e.* the error of the predictor learned from S on a future test case drawn independently from the data distribution. The *probably approximately correct (PAC)* framework has been introduced by Valiant [195] in the context of COLT. Suppose that the predictor computed by the learning method is specified by some parameter $\boldsymbol{\pi}$. A *PAC bound* is a statement of the following form:

$$\Pr_S \{\exists \boldsymbol{\pi} : \mathrm{gen}(\boldsymbol{\pi}) \geq \mathrm{bound}(\boldsymbol{\pi}, S, \delta)\} \leq \delta. \tag{2.33}$$

Here, $\delta \in (0, 1)$ is called *confidence parameter*. The upper bound statistic $\mathrm{bound}(\boldsymbol{\pi}, S, \delta)$ is independent of the data distribution and can be computed from

the sample S. Of course, the particular choice of $\boldsymbol{\pi}$ by the learning method depends on the sample S, and we sometimes write $\mathrm{gen}(S)$ and $\mathrm{bound}(S, \delta)$. The fact that the bound can only be guaranteed to hold with probability $\geq 1 - \delta$ over random draws of S, accounts for the "probably" in PAC. If our learning method chose a predictor independent of S, the *empirical error* $\mathrm{emp}(S) = \mathrm{emp}(\boldsymbol{\pi}, S)$ would converge to $\mathrm{gen}(S)$ almost surely by the law of large numbers and strong large deviation inequalities could be used to obtain the rate of convergence (in fact, large deviation statements are at the heart of PAC theory, and we urge the reader to glance over our exposition in Section A.7). Therefore, it is reasonable to write

$$\mathrm{bound}(\boldsymbol{\pi}, S, \delta) = \mathrm{emp}(\boldsymbol{\pi}, S) + \mathrm{gap}(\boldsymbol{\pi}, S, \delta), \tag{2.34}$$

where the *gap bound term* $\mathrm{gap}(\boldsymbol{\pi}, S, \delta)$ is positive and typically converges to 0 as $n \to \infty$ uniformly over $\boldsymbol{\pi}$. Note that it is unusual but possible that the range of values for $\boldsymbol{\pi}$ (w.r.t. which the $\exists$ is defined in (2.33)) depends on the sample size n or other statistics. In this thesis, we refer to the variable $\mathrm{gen}(\boldsymbol{\pi}) - \mathrm{emp}(\boldsymbol{\pi}, S)$ as *gap*. The "approximately" in PAC accounts for the fact that $\mathrm{gap}(\boldsymbol{\pi}, S, \delta)$ is positive for every sample size n.

Asymptotically, a PAC bound usually implies almost sure convergence[50] of a certain estimator of the generalisation error (*e.g.*, the empirical error) uniformly over internal choices $\boldsymbol{\pi}$ of the algorithm, together with an upper bound on the (uniform) convergence rate, however a PAC statement is somewhat stronger in that it holds (in the PAC sense) for finite sample sizes n. We have already seen that for a "dumb algorithm" which chooses a predictor independent of the sample, large deviation inequalities imply a strong PAC bound. For example, if $p = \mathrm{gen}$, $\hat{p} = \mathrm{emp}(S)$, the Chernoff inequality (Theorem A.3) implies that

$$\Pr_S \left\{ \mathrm{D}_{\mathrm{Ber}}[\hat{p} \,\|\, p] \geq \frac{1}{n} \log \delta^{-1},\ \hat{p} < p \right\} \leq \delta.$$

If the algorithm's choice is restrict to a finite set of values for $\boldsymbol{\pi}$ (sometimes called

[50] Typically, one employs the Borel-Cantelli lemma to show this (e.g., [67], Sect. 7.3).

hypothesis space), of size L say, we can use the *union bound*

$$\Pr\left\{\bigcup_i E_i\right\} \leq \sum_i \Pr\{E_i\} \tag{2.35}$$

in order to obtain a PAC statement:

$$\Pr_S\left\{\mathrm{D}_{\mathrm{Ber}}[\hat{p}\,\|\,p] \geq \frac{1}{n}\left(\log\delta^{-1} + \log L\right),\ \hat{p} < p\right\} \leq \delta.$$

If L is very large or the hypothesis space infinite, the union bound cannot be applied directly, which is where the art of proving PAC bounds begins. For example, the classical VC theory (to be discussed in Section 2.2.3) essentially gives an argument for how the union bound can be applied even in the context of infinite hypothesis spaces and classification problems if these spaces are restricted in a different way.

We now give a rough categorisation of PAC results. Recall the general form from (2.33) and (2.34). If the gap bound term $\mathrm{gap}(\boldsymbol{\pi}, S, \delta)$ is independent of the concrete choice $\boldsymbol{\pi}$ (of course, it will usually depend on characteristics of the hypothesis space), the bound is called *uniform*. In other words, a uniform PAC bound deals with variables

$$\sup_{\boldsymbol{\pi}} \mathrm{gen}(\boldsymbol{\pi}) - \mathrm{emp}(\boldsymbol{\pi}, S),$$

where the supremum is over the hypothesis space. The fixation on the empirical error as only statistic in the bound which depends on the concrete choice of $\boldsymbol{\pi}$ has been motivated above with the law of large numbers. In fact, any statistic which converges almost surely to $\mathrm{gen}(\boldsymbol{\pi})$ for any fixed $\boldsymbol{\pi}$ would do, and a class of PAC bounds indeed focusses on cross-validation error estimates instead of $\mathrm{emp}(\boldsymbol{\pi}, S)$. Historically, the *empirical risk minimisation (ERM)* paradigm which requires to choose a predictor to minimise empirical risk (on the training sample), has been justified by uniform gap bounds for the difference between generalisation and empirical error. While a uniform bound gives the strongest possible statement for a fixed hypothesis class, it is argued in Section 3.1.1 that this restriction on the gap bound term can lead to unnecessarily loose bounds in practice where such a strong statement is often not required. Non-uniform bounds can be tighter in

practice than uniform ones if they incorporate prior knowledge about the problem at hand, and if nature is indifferent in the sense that its chosen data distribution does not "maliciously" contradict these prior assumptions.

If $\text{gap}(\boldsymbol{\pi}, S, \delta)$ does not depend on the concrete sample S (apart from simple statistics such as $n = |S|$), the gap bound is called *data-independent* or *a priori*, as opposed to *data-dependent* or *a posteriori* bounds. This distinction is somewhat artificial, since the empirical error in the bound term depends on S. The potential merits of *a posteriori* bounds are discussed in Section 3.1.1. The notion of non-uniform *a posteriori* bounds has been formalised in the *luckiness framework* of Shawe-Taylor et. al. [177].

The statement of a PAC has to be interpreted with some care.[51] A PAC bound is a joint statement over both the generalisation error and the bound term $\text{bound}(\boldsymbol{\pi}, S, \delta)$. Imagine a large number of statisticians being handed a sample S each drawn independently from the data distribution. Upon training and testing the method on S and evaluating the bound term, for about $1 - \delta$ of them the generalisation error of their method will not be smaller than their bound term value. This does not rule out the possibility that for example the gap could be heavily correlated with the empirical error, being large with high probability whenever $\text{emp}(\boldsymbol{\pi}, S)$ is small. The joint PAC statement could still hold if the probability for a small empirical error is very small, but we happen to observe one with our sample. This possibility cannot be ruled out without truly having access to repeated samples[52] or making assumptions about the data distribution. Phrased differently, a stronger *conditional* statement such as

$$\Pr_S \{\text{gen}(S) \geq \text{bound}(S, \delta) \mid \text{emp}(S) \in I\} \leq \delta$$

for a (non-trivial) interval I chosen *a priori* cannot be directly inferred from a PAC statement.

[51] Thanks to Radford Neal for pointing this out.

[52] In which case it would of course be wasteful not to use them for training.

2.2.2 Concentration Inequalities

The term *concentration inequality* touches a large and diverse field in empirical process theory, from fairly elementary large deviation results over Martingale convergence theorems to deep mathematics. Needless to say, applications of such inequalities reach far beyond learning-theoretical questions, to which such have only recently been applied. We are not in the position to provide more than a brief and incomplete account, borrowing heavily from [106]. The interested reader may consult [199, 100]. We also neglect to cite results properly, see [106] for the original references.

Concentration inequalities can be seen as generalisations of laws of large numbers. The latter state that under mild conditions, the sum of independent variables is close to its expectation with high probability. Here, closeness is measured on the scale of the expectation. This *concentration phenomenon* is true for wide classes of general functions of independent variables. The usefulness of a concentration result is that the study of a complicated random variable, being a function of independent ones, can be replaced by studying the expected value instead which can be much simpler.

One of the simplest concentration results beyond sums is the *Efron-Stein inequality* for the variance of $Z = g(X_1, \ldots, X_n)$, where the X_i are independent variables. Write

$$\mathrm{E}_i Z = \mathrm{E}\left[Z \mid X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n\right].$$

Furthermore, let $\{X_i'\}$ form an independent copy of the sample $\{X_i\}$ (a so-called *ghost sample*), and let $Z_i' = g(X_1, \ldots, X_{i-1}, X_i', X_{i+1}, \ldots, X_n)$. Then the inequality asserts

$$\mathrm{Var}[Z] \leq \sum_{i=1}^{n} \mathrm{E}\left[(Z - \mathrm{E}_i Z)^2\right] = \frac{1}{2} \sum_{i=1}^{n} \mathrm{E}\left[(Z - Z_i')^2\right].$$

A proof can be found in [106]. The proof idea borrows from Martingale theory by considering the Martingale difference sequence

$$V_i = \mathrm{E}[Z \,|\, X_1, \ldots, X_i] - \mathrm{E}[Z \,|\, X_1, \ldots, X_{i-1}].$$

$S_i = \mathrm{E}[Z \,|\, X_1, \ldots, X_i]$ is a *Martingale* (called *Doob's Martingale*, see [67], Chap. 12), i.e. $\mathrm{E}[S_{i+1} \,|\, X_1, \ldots, X_i] = S_i$, thus $\mathrm{E}[V_{i+1} \,|\, X_1, \ldots, X_i] = 0$. Crucially, $Z - \mathrm{E}[Z] = \sum_i V_i$ and $\mathrm{E}[V_i\, V_j] = 0$ for $i < j$, so that

$$\mathrm{Var}[Z] \leq \mathrm{E}\left[\sum_{i=1}^{n} V_i^2\right].$$

Loosely speaking, there is no "cross-talk" between elements of a Martingale difference sequence. The rest of the proof makes use of Jensen's inequality (Theorem A.4) and the convexity of $(\cdot)^2$. Note that the Efron-Stein inequality becomes an equality for Z being the sum of the X_i, so in this sense sums are the least concentrated. The inequality can be weakened further by replacing the $\mathrm{E}_i Z$ by arbitrary (measurable) functions of $X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n$. It is useful in practice if we can guarantee that $Z - Z_i'$ is bounded by c_i (say) almost surely, which is certainly true if the value of g cannot change by more than c_i if only the i-th argument is modified arbitrarily. For example, if all $c_i = 1$, then $\mathrm{Var}[Z] \leq n/2$, thus Z is concentrated around $\mathrm{E}[Z]$ if the latter is of order n.

A much stronger *exponential* concentration inequality can be derived starting from the same *bounded differences* assumption on g, this is known as *Hoeffding's inequality* (e.g., [67], Sect. 12.3). Namely,

$$\Pr\{|Z - \mathrm{E}Z| \geq \varepsilon\} \leq 2\exp\left(-2\varepsilon^2 \Big/ \left(\sum_{i=1}^{n} c_i^2\right)\right).$$

Let $\mathcal{F}_i = \{X_1, \ldots, X_i\}$. Central to the proof is that $\mathrm{E}[V_i \,|\, \mathcal{F}_{i-1}] = 0$ and that conditioned on $\mathcal{F}_{i-1}$ V_i lies within an interval of size c_i. By the convexity of e^{sx} it is not hard to see that

$$\mathrm{E}\left[\exp(s\, V_i) \mid \mathcal{F}_{i-1}\right] \leq e^{(sc_i)^2/8}.$$

We can now use Chernoff's bounding technique (see Section A.7), conditioning on $\mathcal{F}_{i-1}$, $i = n, n-1, \ldots$ in turn and the observation above to finish the proof.

A large number of non-trivial combinatorial problems lead to variables Z with bounded differences. For our purposes, consider

$$Z = \sup_{\boldsymbol{\pi}} \left|\mathrm{emp}(\boldsymbol{\pi}, S) - \mathrm{gen}(\boldsymbol{\pi})\right|. \tag{2.36}$$

Then, Z has bounded differences with $c_i = 1/n$, so is concentrated around its expected value which can be bounded further using for example VC theory (see Section 2.2.3).

Hoeffding's inequality can be improved upon along the same lines as its simple large deviation version by Bernstein's inequality (a relaxation of Chernoff's bound which depends on the average of the variances of X_i and improves upon Hoeffding's bound if the variances are small), although this is technically quite complicated. The rough idea is to impose a *self-bounding property* on g, leading to $\mathrm{Var}[Z] \leq \mathrm{E}Z$. Details of how this can be used to obtain sharper inequalities can be found in [106] and references given there. A whole different avenue is to use information-theoretic concepts and inequalities (such as "conditioning reduces entropy" and the convexity of relative entropy) instead of the decoupling by Martingale arguments. For these and other ideas, see [106].

2.2.3 Vapnik-Chervonenkis Theory

According to Section 2.2.2, the task of proving a PAC result can often be decomposed into showing that the variable quantifying the maximum deviation of generalisation error from bound term (2.36) is concentrated and bounding its expectation. The former can be done using Hoeffding's inequality or variance-dependent refinements. For the latter, the method of Vapnik and Chervonenkis [196] is classical. In this section, we will motivate the ideas behind their work. The binary classification problem can be generalised as follows. Let $X_1, \ldots, X_n$ be i.i.d. variables and $\mathcal{A}$ a set of events A. Denote $\mu(A) = \Pr\{X_1 \in A\}$ and $\hat{\mu}(A) = n^{-1} \sum_i \mathrm{I}_{\{X_i \in A\}}$. The variable of interest is

$$Z = \sup_{A \in \mathcal{A}} |\hat{\mu}(A) - \mu(A)| .$$

It has bounded differences with $c_i = 1/n$, so by Hoeffding:

$$\Pr\{|Z - \mathrm{E}Z| \geq \varepsilon\} \leq 2e^{-2n\varepsilon^2}.$$

VC theory allows to bound EZ in terms of combinatorial characteristics of $\mathcal{A}$ even if the latter is infinite. Define the *shatter coefficient* by

$$S_{\mathcal{A}}(n) = \max_{x_1,\ldots,x_n} |\{\{x_1,\ldots,x_n\} \cap A \mid A \in \mathcal{A}\}|,$$

i.e. the largest number of subsets we can obtain by intersecting an n-set of points with all events in $\mathcal{A}$. In the case we are interested in, A is determined by a classification function, $\mathcal{A}$ is a hypothesis space and x_i contains input and class label, furthermore $x_i \in A$ iff the classifier determined by A makes a mistake on x_i. Then, $S_{\mathcal{A}}(n)$ is the maximum number of different labellings that classifiers from $\mathcal{A}$ can realise on a n-set of input points. $\mathcal{A}$ is the more expressive w.r.t. n-sets, the larger the shatter coefficient. Theorem 1.9 in [105] shows that

$$\mathrm{E}Z \leq 2\sqrt{n^{-1}\log 2S_{\mathcal{A}}(n)}.$$

We do not give the proof here, but only mention the central ideas. We introduce a ghost sample $\{X_i'\}$ as independent copy of $\{X_i\}$. First, by Jensen's inequality (Lemma A.4),

$$\mathrm{E}Z \leq \mathrm{E}\left[\sup_{A\in\mathcal{A}} |\hat{\mu}(A) - \hat{\mu}'(A)|\right],$$

where $\hat{\mu}'$ is the empirical measure for $\{X_i'\}$. The crucial step is *symmetrisation*: introduce Rademacher variables $\sigma_1,\ldots,\sigma_n$, i.i.d. and independent of all others, $\Pr\{\sigma_1 = c\} = 1/2$, $c \in \{-1,+1\}$. Since every pair X_i, X_i' is i.i.d., $c(\mathrm{I}_{\{X_i\in A\}} - \mathrm{I}_{\{X_i'\in A\}})$ has the same distribution for $c = -1$ and $c = +1$, thus

$$\mathrm{E}\left[\sup_{A\in\mathcal{A}} |\hat{\mu}(A) - \hat{\mu}'(A)|\right] = \frac{1}{n}\mathrm{E}\left[\sup_{A\in\mathcal{A}} \left|\sum_{i=1}^{n} \sigma_i \left(\mathrm{I}_{\{X_i\in A\}} - \mathrm{I}_{\{X_i'\in A\}}\right)\right|\right].$$

The purpose of introducing the σ_i is that we can now condition on $\{X_i\}\cup\{X_i'\}$ and concentrate on bounding a variable based on the σ_i only. This is a function of the variables conditioned on, but it can be "worst-case" bounded independent of their values by using the shatter coefficient. The key observation is that the supremum over $\mathcal{A}$ can be replaced by a supremum over a finite *representer set* of all possible intersections of $A \in \mathcal{A}$ with values $\{x_i\} \cup \{x_i'\}$ conditioned on, and the size of this representer set is bounded above by $S_{\mathcal{A}}(2n)$. It is not hard to see

that this together with a large deviation inequality for the Rademacher variables leads to the statement above (see [105] for details). Thus, the central ideas are: introduction of a ghost sample, symmetrisation with Rademacher variables, conditioning, replacing $\mathcal{A}$ by a finite representer set, then essentially using the union bound w.r.t. this set whose size is bounded by the combinatorial shatter coefficient independent of the random variables. In this sense, the VC technique allows the application of the union bound under a combinatorial restriction on the class $\mathcal{A}$. Although $\mathcal{A}$ can be infinite, its expressiveness in terms of how many different classifications it can represent on data is limited. The shatter coefficient is relevant for defining the "dimension" of $\mathcal{A}$, while its size, dimensionality of parameter space, etc. are *not*.

The Shatter coefficient is hard to compute for complicated classes $\mathcal{A}$, but can be bounded in terms of the *VC dimension* V. $S_{\mathcal{A}}(n)$ is bounded by 2^n. If $S_{\mathcal{A}}(n) = 2^n$, we say that $\mathcal{A}$ *shatters* an n-subset of $\mathcal{X}$. As noted above, the shatter coefficient $S_{\mathcal{A}}(n)$ is the maximum number of different labellings $\boldsymbol{y} = (y_i)_i$ we can impose on any n-subset of $\mathcal{X}$ by classifiers in $\mathcal{A}$. Note that if $S_{\mathcal{A}}(n) < 2^n$ that $S_{\mathcal{A}}(m) < 2^m$ for all $m \geq n$, simply because trivially $S_{\mathcal{A}}(n+1) \leq 2\, S_{\mathcal{A}}(n)$. Thus, if there is some n s.t. $S_{\mathcal{A}}(n) < 2^n$, we can define $V \in \mathbb{N}$ as largest value s.t. $S_{\mathcal{A}}(V) = 2^V$, thus the size of the largest subset of $\mathcal{X}$ which can be shattered. Otherwise, set $V = \infty$. As an example, consider a vector space $\mathcal{V}$ of functions $g : \boldsymbol{x} \to \mathbb{R}$ of dimension m and define

$$\mathcal{A} = \{A(g) \,|\, g \in \mathcal{V}\}, \quad A(g) = \{\boldsymbol{x} \,|\, g(\boldsymbol{x}) \geq 0\}.$$

Then, $V = m$ which can be seen as follows. For any $m+1$ points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d+1}$, the embedding of $\mathcal{V}$ into $\mathbb{R}^{m+1}$ via $g \mapsto (g(\boldsymbol{x}_i))_i$ has dimension $\leq m$, so there exists $\boldsymbol{\gamma} \in \mathbb{R}^{m+1} \setminus \{\mathbf{0}\}$ with $\boldsymbol{\gamma}^T (g(\boldsymbol{x}_i))_i = 0$ for all $g \in \mathcal{V}$. W.l.o.g. at least one of the γ_i is negative. Then, the labelling $y_i = 2\mathrm{I}_{\{\gamma_i \geq 0\}} - 1$ cannot be obtained for any $g \in \mathcal{V}$. But we can always find points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ s.t. $\{(g(\boldsymbol{x}_i))_i \,|\, g \in \mathcal{V}\} = \mathbb{R}^m$. Thus, there exist $g_j \in \mathcal{V}$ s.t. $(g_j(\boldsymbol{x}_i))_i = \boldsymbol{\delta}_j$. Then, a labelling $\boldsymbol{y}$ can be achieved by $g = \sum_i y_i g_i$.

If $\mathcal{A}$ has VC dimension $V < \infty$, *Sauer's lemma* states that

$$S_{\mathcal{A}}(n) \leq \sum_{i=0}^{V} \binom{n}{i} \leq \left(\frac{ne}{V}\right)^{V},$$

the second inequality holds only for $n \geq V$. It is now clear that if $V < \infty$, then Z converges to 0 in probability for any data distribution, and the rate of convergence can be bounded in terms of the VC dimension. In this case, $\mathcal{A}$ is sometimes called *PAC learnable*.[53] On the other hand, Vapnik and Chervonenkis showed that if $V = \infty$, there are data distributions for which Z does not converge to 0 in probability. Their results imply a method for deciding the uniform consistency of the ERM paradigm for arbitrary hypothesis spaces $\mathcal{A}$: prove or disprove finiteness of the VC dimension. A detailed account of VC theory and its application to linear classifiers and SVMs is given in [197].

2.2.4 Using PAC Bounds for Model Selection

Suppose we are given a set $\{\mathcal{F}_\alpha\}$ of hypothesis spaces (or models) and we wish to select α with the aim that our method (for example, ERM) restricted to $\mathcal{F}_\alpha$ attains small generalisation error. One reason for pursuing this is that the union $\cup_\alpha \mathcal{F}_\alpha$ may be too large a space to avoid the overfitting problem (see Section A.5) if our method is directly applied to it. In principle, a PAC result which holds for each of the spaces $\mathcal{F}_\alpha$ can be used to select a good α, as has been argued for example in [12]. For a countable set of possible α, we can invoke the multiple testing lemma (essentially the union bound) to show that minimising the PAC bound term plus a log factor accounting for the multiple bound application will select an α close to optimal for large n, see [12] for details. However, it is rather less clear whether model selection via PAC bounds is justified for the rather *small* sample sizes often encountered in practice, for which common PAC results frequently cannot give tight guarantees for most values of α.

Define the *slack* in a bound as the difference between bound value and generalisation error. The arguments in [12] are certainly plausible. However, in some

[53] If we ignore issues of computational complexity.

COLT publications the virtue of PAC bounds for model selection is suggested for sample sizes n for which the slack is obviously an order of magnitude above the generalisation error itself. This is often done by plotting bound values and generalisation error estimates from independent data as curves of α,[54] then noting that the respective minimum points are somewhat close (although the corresponding minima are not). This can be regarded as an empirical observation, in the same way as other techniques like cross-validation or marginal likelihood maximisation are often observed to work fine in practice, but not as much more. An example is the popular SVM which is often claimed to be "theoretically justified" by VC theorems. However, most of the known bounds for SVM are trivial for the region of sample sizes for which the SVM algorithm can feasibly be run, and model selection in practice is typically not done by minimising these bounds.

Model selection by bound minimisation in such a case makes sense only if the slack can be shown to be of much smaller variability (w.r.t. different values of α) than the generalisation error itself. If this were true, then we could infer that the corresponding minimum points are close. However, it is likely that proving such robustness of the slack against changing α will be very much harder to show than a PAC result itself in all but toy situations, or simply does not hold. We are not aware of any results in this direction. In order to prove a modern PAC result, typically a host of bounding steps based on very different mathematical ideas is applied and contribute to the slack. While the bound term usually represents a trade-off between good fit to S and complexity of the predictor, the complexity penalty and the weighting between these terms strongly depends on the bounding techniques used. It seems very unlikely to us that the resulting slack should not depend significantly on α itself.

[54] This practice also neglects the multiple testing issue, which does not allow to apply the bound *simultaneously* to different values of α without accounting for this somehow, but that is not the point here.

Chapter 3

PAC-Bayesian Bounds for Gaussian Process Methods

The PAC-Bayesian theorem of McAllester is an unusual result in the field of statistical learning theory. While other theorems employ heavy concepts from empirical process theory and beyond, the PAC-Bayesian theorem can be proved without much effort, using simple properties of convex functions (one of the contributions of this chapter is to give such a simple proof). Nevertheless, McAllester's result is very powerful, applicable to a large number of Bayes-like learning schemes and highly configurable to available task prior knowledge, characteristics which most other PAC bounds lack to that extent. Maybe because of its simple and direct proof, the PAC-Bayesian theorem can also be very tight in practically relevant situations when many other "heavy-weight" theorems struggle to give non-trivial guarantees. In this chapter, we present various extensions of this remarkable result, along with a simple proof which points out convex (Legendre) duality as core principle, in contrast to most other recent PAC results which are based on the union bound and combinatorics. We then apply the theorem to approximate Bayesian Gaussian process classification, obtaining very tight bounds as judged by experimental studies. For fairly small sample sizes, the results are highly non-trivial and outperform other recent kernel classifier bounds we compared against by a wide margin.

The structure of this chapter is as follows. Section 3.1 is introductory and stresses the need in practice for bounds which are strongly dependent on learning method, training sample and task prior knowledge. In Section 3.2 we introduce several variants of the PAC-Bayesian theorem along with a proof and consider some extensions. These theorems are applied to Gaussian process classification in Section 3.3. In Section 3.4, we mention related work, and in Section 3.5 we present experimental results on a handwritten digits recognition task. The chapter is closed by the discussion in Section 3.6.

Parts of the results presented here have been published previously, as is detailed in Section 1.1.

3.1 Data-dependent PAC Bounds and PAC-Bayesian Theorems

In this section, we discuss a number of shortcomings of classical Vapnik-Chervonenkis generalisation error bounds w.r.t. tightness in practical applications and show directions for improvement. We introduce a number of Bayesian types of classifiers. Finally, PAC-Bayesian theorems are motivated as a way to address the shortcomings of the classical results.

3.1.1 The Need for Data-dependent Bounds

Recall the binary classification problem introduced in Section A.5, where the training sample is $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\},\ \boldsymbol{x}_i \in \mathcal{X},\ y_i \in \{-1, +1\}$. Also recall the notion of a PAC bound on the generalisation error introduced in Section 2.2.1, which provides an estimated upper bound as a function of the training sample S and a confidence parameter δ, such that the event of the bound being violated has probability less than δ over random draws of S. Useful PAC bounds can only be proved in practically relevant situations if the complexity of the range of classification functions in response to finite training data is limited, e.g. via regularisation (see Section A.5). Without such limitation, the problem of learning

from finite data is ill-posed.

The ideas behind classical VC bounds have been introduced in Section 2.2.3. It is important to note that the classical theory has not necessarily been developed with practical applications in mind, but rather to answer the question under which conditions the principle of empirical risk minimisation (ERM) over a hypothesis space $\mathcal{H}$ is uniformly consistent. This question is solved completely by VC theory: $\mathcal{H}$ is "learnable" iff its VC dimension is finite. Only much later have these results been linked with practically successful schemes such as the SVM (see Section 2.1.6). However, once applied to real-world situations, reasonable samples size and non-trivial models, classical VC bounds as well as many more recent developments have low or zero impact, because the bound values typically lie orders of magnitude above the truth.

Theoreticians tend to brush away such objections by citing minimax lower bounds which match the VC upper bounds fairly closely asymptotically. This argument holds up if we think of a PAC bound as a statement which is written down once and is then blindly and uniformly applied to any statistical problem at hand. But it is misleading in the context of real-world statistics, because the setup of these lower bounds is very different from the PAC scenario. Although a PAC bound holds for all data distributions, its value can depend strongly on the data distribution. Modern bounds can be orders of magnitude smaller if the data distribution corresponds to the prior assumptions they encode than if it is constructed in a malicious way. Practitioners may not be too concerned with this issue, after all most of statistics and natural science would not work with nature as a malicious data distribution constructor! But exactly this is the setup of minimax lower bounds: given a particular statistical method, there are always some distributions which lead to very slow learning. However, these worst-case distributions are very unusual[1] and constructed in order to exploit weaknesses of the learning method.

Note that many researchers are more interested in bounding the rate of convergence of the gap as $n \to \infty$ than in precise gap bounds. The problem with this

[1]They exhibit characteristics which typically make our model of them (on which a learning method is based) completely useless.

is that some insist the rate bounds have to be independent of the data distribution, thus to be the same for pathetic worst-case distributions than for any other more sensible distribution (in light of the model used). In order to obtain useful rate bounds in general (e.g., to show consistency) the model has to be restricted or regularised appropriately, and ideally theoretical results should guide these limiting choices in practice. But if a result does not depend on the true source, it will in general not support the single most important principle in statistics: obtain as much information as possible about the task and adjust the model to be compatible with this information. Rather, it will suggest to hedge against worst-case scenarios by choosing unrealistically simple models. In fact, it will typically be indifferent to whether we try to encode prior information faithfully or not, or even worse to vote against such efforts because they might increase our method's vulnerability to worst-case scenarios.

Why is it possible to improve on classical VC results in practice? Recall that the *gap* is the difference between generalisation error and empirical error. A classical VC theorem will typically be based on a hypothesis space $\mathcal{H}$ of finite VC dimension and will bound the gap for the worst choice of $h \in \mathcal{H}$. This bound will then of course hold for the particular algorithm we really use, but at the same time it holds just as well for any other method of choosing from $\mathcal{H}$, even for the "maximally malicious" algorithm which knows the data distribution perfectly and selects a hypothesis with maximal gap. This may be overly ambitious: in the words of Vapnik, *when solving a given problem, one should avoid solving a more general problem as an intermediate step* [197]. Sometimes it is possible to shift particularities of the algorithm into the definition of $\mathcal{H}$, but a better solution is to consider complexity measures other than the VC dimension (or scale-sensitive versions thereof) which are specific to the algorithm used. This can also alleviate another problem with classical VC bounds, namely that the gap bound depends on the sample S only via its size n. Recall from Section 2.2.1 that such bounds are called data-independent or *a priori*, in contrast to data-dependent or *a posteriori* bounds. Since for any fixed predictor, the empirical error converges against the true one almost surely, the empirical error is certainly

a major "ingredient" for any bound expression, but the limitation to this one and only statistic is sensible only in the classical setting where ERM alone is to be analysed. In the bounds we are interested in here, additional statistics are used to drive data-dependent complexity measures, potentially using more information in the sample than merely the empirical error and the size. Finally, classical VC bounds are restricted in how prior knowledge about the task might be encoded in the bound. This is possible to some degree, by creating a hierarchy[2] of nested hypothesis spaces of growing VC dimension corresponding to a prior based on Occam's razor, but the process is very complicated in practice.

To summarise, the VC dimension of a hypothesis space seems unsuitable as a complexity measure in practice. It is neither flexible nor fine-grained enough and can hardly be adjusted to algorithms, models and prior knowledge. It also does not depend on the sample S. These shortcomings have been mentioned before, and the *luckiness framework* [177] has been proposed as an alternative. The PAC-Bayesian bounds to be discussed in this chapter can be seen as very strong realisations of this framework, but we will present them within the formally much simpler and more established framework of Bayesian inference. It is important to point out that we will not compromise the basic validity of PAC statements in any way:

- In order to construct a classification method *and* a data-dependent bound for it, we follow *Bayesian modelling assumptions* (or other heuristics): available prior knowledge is encoded, within feasibility constraints, into a probabilistic model and prior distributions, or an algorithm for prediction is derived in a different heuristic way. A distribution-free bound, which will in general depend on the algorithm, the prior assumptions and the sample S (beyond just the empirical error) is used to bound the generalisation error. The extent to which the unknown data distribution is compatible with these assumptions will in general determine the accuracy of the method *and* the observed tightness of the bound, but it does *not* compromise the

[2]This *structural risk minimisation* approach is used to deal with hypothesis spaces of infinite VC dimension, such as for example SVMs in a feature space.

validity of the theorem.

- The statement of the bound holds under standard *PAC assumptions*: We are given an i.i.d. training sample S from the data distribution which is otherwise *completely unknown*. Whether the data distribution is in agreement with the prior assumptions or violates them, does *not* influence the validity of the statement.

3.1.2 Bayesian Classifiers. PAC-Bayesian Theorems

Recall the setup of the binary classification model defined in Section A.6.1, based on a family $\{u(\cdot \,|\, \boldsymbol{w})\}$ parameterised by $\boldsymbol{w}$ and a noise distribution $P(y|u)$. We assume that the latter has the form $P(y|u) = f(y(u+b))$, where b is a bias hyperparameter, and f is symmetric around $(0, 1/2)$: $f(-x) = 1 - f(x)$. Note that $\boldsymbol{w}$ need not be a finite-dimensional vector. For example, in our application to non-parametric models below we will identify $\boldsymbol{w}$ with the process $u(\cdot \,|\, \boldsymbol{w})$ itself. A Bayesian analysis for this model (see Section A.6.2) requires the specification of a prior distribution $P(\boldsymbol{w})$. If $Q(\boldsymbol{w})$ is the posterior for the training sample S, the target probability for a new point $\boldsymbol{x}_*$ is predicted as $Q(y_*|\boldsymbol{x}_*, S) = \mathrm{E}_Q[P(y_* \,|\, u(\boldsymbol{x}_*|\boldsymbol{w}))]$, and the *predictive classifier* is $\mathrm{sgn}(Q(y_* = +1|\boldsymbol{x}_*, S) - 1/2)$.

A number of related classifiers have been studied. The *Bayes classifier* predicts $y_{\mathrm{Bayes}}(\boldsymbol{x}_*) = \mathrm{sgn}(\mathrm{E}_Q[u(\boldsymbol{x}_*|\boldsymbol{w})] + b)$, while the *Bayes voting classifier* outputs $y_{\mathrm{Vote}}(\boldsymbol{x}_*) = \mathrm{sgn}\,\mathrm{E}_Q[\mathrm{sgn}(u(\boldsymbol{x}_*|\boldsymbol{w}) + b)]$. Note that our terminology is non-standard here: some authors would refer to our predictive classifier as Bayes classifier, while others use the term "Bayes classifier" to denote the optimal classifier for the data distribution. In general, all three classifiers (predictive, Bayes, Bayes voting) are different, but if the distribution of $u(\boldsymbol{x}_*|\boldsymbol{w})$, $\boldsymbol{w} \sim Q$ is symmetric around its mean for every $\boldsymbol{x}_*$, then they all agree. Another type of classifier, called *Gibbs classifier*, has been studied in learning theory (e.g., [70]). Given a test point $\boldsymbol{x}_*$, it predicts the corresponding target by first sampling $\boldsymbol{w} \sim Q$, then returning $y_* = \mathrm{sgn}(u(\boldsymbol{x}_*|\boldsymbol{w}) + b)$, plugging in the sampled parameter vector. Note

that a Gibbs classifier has a probabilistic element and requires coin tosses for prediction. Note also that if the targets of several test points are to be predicted, the parameter vectors sampled for this purpose are independent.[3] In the situations we are interested in practice, the Gibbs classifier often performs somewhat worse than the corresponding Bayes classifier. We will discuss their relationship in more detail in Section 3.2.5. The Gibbs classifier is the method of choice if for some reason we are restricted to use a single $u(\boldsymbol{x}_*|\boldsymbol{w})$ for prediction.

In [116, 115, 117], McAllester proved a number of *PAC-Bayesian theorems* applicable to Gibbs classifiers. In general, a PAC-Bayesian theorem is simply a PAC bound which deals with Bayes-like classifiers constructed based on expectations over hypotheses or discriminants with respect to a posterior distribution Q. It is important to note that Q need not be a Bayesian posterior distribution for some model, but can be chosen by the learning algorithm at will. McAllester's theorem incorporates all directions of improvement mentioned in Section 3.1.1 and suggests a new complexity measure which is data and algorithm-dependent and can be adjusted based on prior knowledge. More importantly, the measure is compatible with the aims of Bayesian modelling and prior assessment, so that the theorem is especially suitable for applications to (approximate) Bayesian algorithms. In the following section, we will present the theorem and a range of extensions, together with a simple and intuitive proof.

3.2 The PAC-Bayesian Theorem for Gibbs Classifiers

In this section, we present and prove a range of PAC-Bayesian theorems for Gibbs classifiers, both for the binary classification problem and more general multi-class scenarios or arbitrary bounded loss functions. A simple extension to binary Bayes

[3] Readers familiar with *Markov chain Monte Carlo* methods (see Section A.6.3) will note the similarity with a MCMC approximation (based on one sample of $\boldsymbol{w}$ only) of the corresponding Bayes classifier for $Q(\boldsymbol{w})$. The difference is that typically in MCMC, the sample representing the posterior $Q(\boldsymbol{w})$ is retained and used for many predictions, while in the Gibbs classifier, we use each posterior sample only once.

classifiers is motivated as well.

3.2.1 The Binary Classification Case

In this section, we present McAllester's PAC-Bayesian theorem for binary classification. The theorem deals with a Gibbs classifier (see Section 3.1.2) whose mixture distribution $Q(\boldsymbol{w})$ may depend on the training sample S, which is why $Q(\boldsymbol{w})$ is referred to as "posterior distribution". In order to eliminate the probabilistic element in the Gibbs classifier itself, the bound is on the gap between expected generalisation error and expected empirical error, where the expectation is over $Q(\boldsymbol{w})$. The theorem can be configured by a prior distribution $P(\boldsymbol{w})$ over parameter vectors. The gap bound term depends strongly on the relative entropy (Definition A.4) $\mathrm{D}[Q \,\|\, P]$ between $Q(\boldsymbol{w})$ and the prior $P(\boldsymbol{w})$. Here, we assume that $Q(\boldsymbol{w})$ and $P(\boldsymbol{w})$ are absolutely continuous w.r.t. some positive measure. Recall the Bernoulli relative entropy $\mathrm{D}_{\mathrm{Ber}}[q \,\|\, p]$ from (A.7). It is convex in (q, p), furthermore $\mathrm{D}_{\mathrm{Ber}}[q \,\|\, \cdot]$ is strictly decreasing for $p < q$, strictly increasing for $p > q$. Thus, the following mapping

$$\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon) = [p_L, p_U] \;\text{ s.t. }\; \mathrm{D}_{\mathrm{Ber}}[q \,\|\, p_L] = \mathrm{D}_{\mathrm{Ber}}[q \,\|\, p_U] = \varepsilon,\; p_L \leq q,\; p_U \geq q, \quad (3.1)$$

is well-defined for $q \in (0, 1)$ and $\varepsilon \geq 0$. We also define $\mathrm{D}_{\mathrm{Ber}}^{-1}(0, \varepsilon) = [0, 1 - e^{-\varepsilon}]$ and $\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \infty) = [0, 1]$. $\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon)$ can be seen as "relative entropy ball" of radius ε around q. Note that due to the convexity of $\mathrm{D}_{\mathrm{Ber}}$, we can compute the interval limits of $\mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon)$ easily using Newton's algorithm. It is clear by definition that for $\varepsilon \geq 0,\; p \in [0, 1]$:

$$p \in \mathrm{D}_{\mathrm{Ber}}^{-1}(q, \varepsilon) \iff \mathrm{D}_{\mathrm{Ber}}[q \,\|\, p] \leq \varepsilon. \quad (3.2)$$

If $\delta \in (0, 1)$ is a confidence parameter, we have the following result.

Theorem 3.1 (PAC-Bayesian theorem [115]) *For any data distribution over $\mathcal{X} \times \{-1, +1\}$, we have that the following bound holds, where the probability is over random i.i.d. samples $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\}$ of size n drawn from the data distribution:*

$$\Pr_S \left\{ \mathrm{gen}(Q) \in \mathrm{D}_{\mathrm{Ber}}^{-1}(\mathrm{emp}(S, Q), \varepsilon(\delta, n, P, Q)) \;\text{ for all } Q \right\} \geq 1 - \delta. \quad (3.3)$$

Here, $Q = Q(\boldsymbol{w})$ is an arbitrary "posterior" distribution over parameter vectors, which may depend on the sample S and on the prior P. Furthermore,

$$\begin{aligned}
\mathrm{emp}(S, Q) &= \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\frac{1}{n}\sum_{i=1}^{n} \mathrm{I}_{\{\mathrm{sgn}(u(\boldsymbol{x}_i \,|\, \boldsymbol{w})+b)\neq y_i\}}\right], \\
\mathrm{gen}(Q) &= \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\mathrm{E}_{(\boldsymbol{x}_*, y_*)}\left[\mathrm{I}_{\{\mathrm{sgn}(u(\boldsymbol{x}_* \,|\, \boldsymbol{w})+b)\neq y_*\}}\right]\right], \\
\varepsilon(\delta, n, P, Q) &= \frac{1}{n}\left(\mathrm{D}[Q \,\|\, P] + \log\frac{n+1}{\delta}\right).
\end{aligned}$$

Here, $\mathrm{emp}(S, Q)$ is the expected empirical error, $\mathrm{gen}(Q)$ the expected generalisation error of the Gibbs classifier based on $Q(\boldsymbol{w})$ (note that the probability in $\mathrm{gen}(Q)$ is over $(\boldsymbol{x}_*, y_*)$ drawn from the data distribution, independently of the sample S).

Although $\mathrm{D}_{\mathrm{Ber}}^{-1}$ is easy to compute, it may be awkward to use in certain situations. Therefore, $\mathrm{D}_{\mathrm{Ber}}$ is frequently approximated by lower bounds which are fairly tight if $q \approx p$. If $p \geq q$, we have $\mathrm{D}_{\mathrm{Ber}}[q \,\|\, p] \geq (p-q)^2/(2p)$ which can be seen by taking derivatives of both sides w.r.t. q. It follows that if $p \geq q$ and $\mathrm{D}_{\mathrm{Ber}}[q \,\|\, p] \leq \varepsilon$, then

$$p \leq q + 2\varepsilon + \sqrt{2\varepsilon q},$$

leading to

$$\mathrm{gen}(Q) \leq \mathrm{emp}(S, Q) + 2\varepsilon + \sqrt{2\varepsilon \mathrm{emp}(S, Q)},$$

which shows that the gap bound scales roughly as $2\mathrm{D}[Q \,\|\, P]/n$ if the empirical error is small. Needless to say, the use of this additional lower bound is not recommended in practice.

Note that McAllester's theorem applies more generally to bounded loss functions and makes use of Hoeffding's inequality for bounded variables. A further generalisation is shown in Section 3.2.4. However, for the special case of zero-one loss, we can use techniques tailored for binomial variables which give considerably tighter results than Hoeffding's bound if the expected empirical error of the Gibbs classifier is small. The theorem can be generalised in various ways. In Section 3.2.2, we present a multi-class version which encompasses the binary classification case, and the proof given there will serve to prove Theorem 3.1. A

slightly shorter direct proof can be found in [170]. Complexity measures other than the relative entropy may be considered, as discussed in Section 3.2.6.1. Finally, a simple but rather weak extension to the Bayes classifier is given in Section 3.2.5.

We should stress once more, in line with Section 3.1.1, that the PAC-Bayesian theorem does *not* require the true data distribution to be constrained in any way depending on the prior $P(\boldsymbol{w})$ and the model class. For example, other analyses (which are not PAC) try to characterise *learning curves*: given that S has been generated from the given model and prior, they analyse the generalisation error of a Gibbs or Bayes classifier, averaged over the data distribution (e.g., [71, 186]). Or, given that S has been generated i.i.d. from a fixed member $\boldsymbol{w}_0$ of the family, they derive the convergence rate of some distance between this (product) data distribution and the Bayesian marginal likelihood $\mathrm{E}_{P(\boldsymbol{w})}[P(S|\boldsymbol{w})]$ [30]. It is clear that under such more restrictive assumptions to start with, stronger results can be achieved than the PAC-Bayesian theorem.

3.2.2 Confusion Distributions and Multiple Classes

In practice, many classification problems come with more than two classes. Although we can always tackle such problems using a sufficient number of binary classifiers, it is more principled and data-economic to instead use a multi-class model (see Section 2.1.2). In this subsection, we assume that the targets to be predicted are class labels from $\{1, \ldots, C\}$, $C \geq 2$. Probabilistic rules mapping input points $\boldsymbol{x}$ to distributions over $\{1, \ldots, C\}$ will also be considered. A rule $\boldsymbol{\pi}(\cdot\,|\boldsymbol{w})$ is used to predict y_* at a test point $\boldsymbol{x}_*$ by sampling it from the distribution $\boldsymbol{\pi}(\boldsymbol{x}_*|\boldsymbol{w})$ over $\{1, \ldots, C\}$. By restricting ourselves to delta distributions $\boldsymbol{\pi}(\cdot\,|\boldsymbol{w})$, we can always recover the special case of purely deterministic rules.

The application of a PAC bound is really only a statistical test and as such has to be embedded into the experimental design. Even in the binary case ($C = 2$), we might be interested in bounding other aspects of the data distribution than the generalisation error, e.g. the probability of false positives, and as C grows so does the number of possible questions which can be tested based

on the training sample. Especially in the multi-class case, a useful PAC bound should support individual designs concerned with more general properties than the generalisation error, including the latter as a special case. It is not hard to extend the PAC-Bayesian theorem in this respect. We capture the notion of probabilistic rules by introducing a variable $\boldsymbol{r} \sim R(\boldsymbol{r})$. $\boldsymbol{r}$ is the source of random coins required to sample $y_* \sim \boldsymbol{\pi}(\boldsymbol{x}_*|\boldsymbol{w})$. It is independent of all other variables, and R is fixed and independent of all other distributions. Effectively, $\pi_{y_*}(\boldsymbol{x}_*|\boldsymbol{w}) = \Pr_{\boldsymbol{r}\sim R}\{y(\boldsymbol{x}_*|\boldsymbol{w}, \boldsymbol{r}) = y_*\}$ for deterministic rules $y(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{r})$ mapping into $\{1, \ldots, C\}$. Note that the corresponding Gibbs rule based on the distribution $Q(\boldsymbol{w})$ is evaluated at $\boldsymbol{x}_*$ by sampling $\boldsymbol{w} \sim Q$, $\boldsymbol{r} \sim R$ independently, then outputting $y(\boldsymbol{x}_*|\boldsymbol{w}, \boldsymbol{r})$. The experimental design now implies a finite set $\mathcal{L}$ of size L and a distribution $\boldsymbol{p}$ over $\mathcal{L}$ which is related to the unknown data distribution and the posterior distribution $Q(\boldsymbol{w})$ as follows: if $(\boldsymbol{x}_*, y_*)$ is sampled from the data distribution, $\boldsymbol{w} \sim Q$ and $\boldsymbol{r} \sim R$ independently, then the variable $l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r})$ has distribution $\boldsymbol{p}$, where l is a known function. For example, if $\mathcal{L} = \{0, 1\}$ and $l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r}) = \mathrm{I}_{\{y(\boldsymbol{x}_*|\boldsymbol{w},\boldsymbol{r})\neq y_*\}}$, then $\boldsymbol{p} = (1 - e_{\mathrm{Gibbs}}, e_{\mathrm{Gibbs}})$, where $e_{\mathrm{Gibbs}} = \Pr\{y(\boldsymbol{x}_*|\boldsymbol{w}, \boldsymbol{r}) \neq y_*\}$ is the expected generalisation error of the Gibbs rule. More generally, we may be interested in the *joint confusion distribution*

$$F(y_*, \tilde{y}) = \mathrm{E}_{\boldsymbol{x}_*}\left[P(y_*|\boldsymbol{x}_*)\mathrm{E}_{\boldsymbol{w}\sim Q}\left[\pi_{\tilde{y}}(\boldsymbol{x}_*|\boldsymbol{w})\right]\right], \tag{3.4}$$

where $P(y_*|\boldsymbol{x}_*)$ denotes the conditional data distribution, for example in order to bound the probabilities of false positives and true negatives in the case of binary classification. Now, if $\mathcal{L} = \{1, \ldots, C\} \times \{1, \ldots, C\}$ and $l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r}) = (y_*, y(\boldsymbol{x}_*|\boldsymbol{w}, \boldsymbol{r}))$, then $\boldsymbol{p}$ coincides with the joint confusion distribution F. In short, a general PAC bound allows us to make inferences about the components of $\boldsymbol{p}$ based on the observed sample S.

We can now state the following generalisation of the PAC-Bayesian Theorem 3.1 for binary classification. Suppose we are given a set $\mathcal{L}$ of size L and a mapping $l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r})$ into $\mathcal{L}$, furthermore an arbitrary prior distribution $P(\boldsymbol{w})$ over parameter vectors, and we choose a confidence parameter $\delta \in (0, 1)$. Then, the following result holds.

Theorem 3.2 (Extended PAC-Bayesian Theorem) *For any data distribution over $\mathcal{X} \times \{-1, +1\}$, we have that the following bound holds, where the probability is over random i.i.d. samples $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\}$ of size n drawn from the data distribution:*

$$\Pr_S \left\{ \mathrm{D}\left[\hat{\boldsymbol{p}}(S, Q) \,\|\, \boldsymbol{p}(Q)\right] \leq \varepsilon(\delta, n, P, Q) \text{ for all } Q \right\} > 1 - \delta. \tag{3.5}$$

Here, $Q = Q(\boldsymbol{w})$ is an arbitrary "posterior" distribution over parameter vectors, which may depend on the sample S and on the prior P. Furthermore, $\boldsymbol{p}(Q)$ is a distribution over $\mathcal{L}$, induced from the data distribution as follows:

$$\left[\boldsymbol{p}(Q)\right]_{l_*} = \Pr_{(\boldsymbol{x}_*, y_*), \boldsymbol{w}, \boldsymbol{r}} \left\{ l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r}) = l_* \right\},$$

where $(\boldsymbol{x}_*, y_*)$ is drawn from the data distribution, $\boldsymbol{w} \sim Q$ and $\boldsymbol{r} \sim R$, all independently. $\hat{\boldsymbol{p}}(S, Q)$ is an empirical estimate of $\boldsymbol{p}(Q)$ given by

$$\left[\hat{\boldsymbol{p}}(S, Q)\right]_{l_*} = \frac{1}{n} \sum_{i=1}^{n} \Pr_{\boldsymbol{w}, \boldsymbol{r}} \left\{ l(\boldsymbol{x}_i, y_i, \boldsymbol{w}, \boldsymbol{r}) = l_* \right\}.$$

Furthermore,

$$\varepsilon(\delta, n, P, Q) = \frac{\mathrm{D}[Q \,\|\, P] + (L - 1)\log(n + 1) - \log \delta}{n}.$$

The proof of this theorem is given below in this section. Note that Theorem 3.1 is a special case of Theorem 3.2: it is obtained if we set $\mathcal{L} = \{0, 1\}$ and $l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r}) = \mathrm{I}_{\{y(\boldsymbol{x}_* | \boldsymbol{w}, \boldsymbol{r}) \neq y_*\}}$ with $y(\boldsymbol{x}_* | \boldsymbol{w}, \boldsymbol{r}) = \mathrm{sgn}(u(\boldsymbol{x}_* | \boldsymbol{w}) + b)$.

The theorem renders a level ε such that with high confidence $1 - \delta$ we have $\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}] \leq \varepsilon$, where $\hat{\boldsymbol{p}} = \hat{\boldsymbol{p}}(S, Q)$, $\boldsymbol{p} = \boldsymbol{p}(Q)$ and $\varepsilon = \varepsilon(\delta, n, P, Q)$. In other words, the unknown vector $\boldsymbol{p} \in \mathbb{R}^L$ lies in the closed convex set

$$\mathcal{P}(\hat{\boldsymbol{p}}, \varepsilon) = \left\{ \boldsymbol{q} \,\middle|\, \boldsymbol{q} \geq 0,\ \mathbf{1}^T \boldsymbol{q} = 1,\ \mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{q}] \leq \varepsilon \right\}. \tag{3.6}$$

$\mathcal{P}(\hat{\boldsymbol{p}}, \varepsilon)$ can be seen as "relative entropy ball" of radius ε around the centre $\hat{\boldsymbol{p}}$, a multidimensional generalisation of $\mathrm{D}_{\mathrm{Ber}}^{-1}$ (see Equation 3.1). By cutting this ball with planes, we can derive bounds on projections $\boldsymbol{c}^T \boldsymbol{p}$. In Appendix B.1 we provide an explicit example for how the theorem can be used in practice.

As one of the major contributions of this chapter, we present a proof of Theorem 3.2. McAllester's theorem [117] is a special case of this theorem, yet our proof is considerably simpler than the original one and leads to several possible avenues of generalisation. Our bound is also tighter in the special case of binary classification.

Recall the notation introduced further above in this section. For notational simplicity, we define $\tilde{\boldsymbol{w}} = (\boldsymbol{w}, \boldsymbol{r})$, thus pairing the two possible random sources of the randomised Gibbs rule.[4] We also extend prior P and posterior Q by setting $dP(\tilde{\boldsymbol{w}}) = dP(\boldsymbol{w})\, dR(\boldsymbol{r})$, $dQ(\tilde{\boldsymbol{w}}) = dQ(\boldsymbol{w})\, dR(\boldsymbol{r})$ (product measures). Define

$$[\boldsymbol{p}(\tilde{\boldsymbol{w}})]_{l_*} = \mathrm{E}_{(\boldsymbol{x}_*, y_*)} \left[\mathrm{I}_{\{l(\boldsymbol{x}_*, y_*, \tilde{\boldsymbol{w}}) = l_*\}} \right], \quad [\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})]_{l_*} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{I}_{\{l(\boldsymbol{x}_i, y_i, \tilde{\boldsymbol{w}}) = l_*\}}, \quad l_* \in \mathcal{L},$$

where the expectation is over the unknown data distribution, and the sample is $S = \{(\boldsymbol{x}_i, y_i) \,|\, i = 1, \ldots, n\}$. Let $\Delta(\tilde{\boldsymbol{w}}) = \mathrm{D}[\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}}) \,\|\, \boldsymbol{p}(\tilde{\boldsymbol{w}})]$. $\Delta(\tilde{\boldsymbol{w}})$ simply measures, for a fixed instance $\tilde{\boldsymbol{w}}$ of a fixed rule, the divergence between $\boldsymbol{p}(\tilde{\boldsymbol{w}})$ and its empirical estimate $\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})$ in a convenient way.

Fix $\tilde{\boldsymbol{w}}$, and write $\hat{\boldsymbol{p}} = \hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})$, $\boldsymbol{p} = \boldsymbol{p}(\tilde{\boldsymbol{w}})$. Since $\tilde{\boldsymbol{w}}$ is fixed independent of the sample S, the strong law of large numbers asserts that $\hat{\boldsymbol{p}}$ converges against $\boldsymbol{p}$ almost surely and we can derive a strong large deviation inequality for $\Delta(\tilde{\boldsymbol{w}})$. We may expect that this guarantee remains valid if instead of fixing $\tilde{\boldsymbol{w}}$ *a priori*, we sample it from the prior distribution $P(\tilde{\boldsymbol{w}})$, because P does not depend on the sample S either. The first part of the proof makes this argument sound. The reason for our *particular* choice of $\Delta(\tilde{\boldsymbol{w}})$ is that the corresponding large deviation inequality is tight and can be proved easily. For fixed $\tilde{\boldsymbol{w}}$, $n\,\hat{\boldsymbol{p}}$ is multinomial $(n, \boldsymbol{p})$ distributed. Csiszár and Körner [42] refer to $\hat{\boldsymbol{p}}$ as the *type* of the underlying i.i.d. sequence $\{l(\boldsymbol{x}_i, y_i, \tilde{\boldsymbol{w}}) \,|\, i = 1, \ldots, n\}$, and we will use their elegant method of types (see also [34], Sect. 12.1) for the first part of the proof. As is shown in Appendix B.2, we have

$$\mathrm{E}_S \left[e^{n \mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}]} \right] \leq (n+1)^{L-1}. \tag{3.7}$$

[4] If the rules (parameterised by $\boldsymbol{w}$) are deterministic, we set $\tilde{\boldsymbol{w}} = \boldsymbol{w}$ and forget about $\boldsymbol{r}$ altogether.

Now, taking the average over $\tilde{\boldsymbol{w}} \sim P$ and using Markov's inequality (Theorem A.1), we obtain

$$\Pr_S \left\{ \mathrm{E}_{\tilde{\boldsymbol{w}} \sim P} \left[e^{n\,\Delta(\tilde{\boldsymbol{w}})} \right] > \frac{(n+1)^{L-1}}{\delta} \right\} \leq \delta. \tag{3.8}$$

It is of course essential here that P does not depend on the sample S. In other words, it is easy to prove strong large deviation bounds for "dumb" Gibbs rules which do not depend on the training sample! For example, we can use the concavity of log and the convexity of $\Delta(\tilde{\boldsymbol{w}})$ together with Jensen's inequality (Lemma A.4) to see that

$$\mathrm{D}\left[\mathrm{E}_{\tilde{\boldsymbol{w}} \sim P}[\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})] \,\middle\|\, \mathrm{E}_{\tilde{\boldsymbol{w}} \sim P}[\boldsymbol{p}(\tilde{\boldsymbol{w}})] \right] \leq \frac{(L-1)\log(n+1) - \log\delta}{n} \tag{3.9}$$

with probability at least $1-\delta$ over random draws of S. Unfortunately, this is also quite uninteresting in practice. The cornerstone of the PAC-Bayesian theorem is a *generic* way of converting such bounds on "dumb" *a priori* Gibbs rules into useful bounds on *a posteriori* rules (the same method can be applied to Bayes rules as well, see Section 3.4.1).

Let us take the statement (3.8) for the "dumb" Gibbs rule based on the prior P and ask what happens if we replace P against the posterior Q. Fix an arbitrary sample S for which indeed

$$\mathrm{E}_{\tilde{\boldsymbol{w}} \sim P} \left[e^{n\,\Delta(\tilde{\boldsymbol{w}})} \right] \leq K, \quad K = \frac{(n+1)^{L-1}}{\delta}. \tag{3.10}$$

If we can show that

$$\mathrm{E}_{\tilde{\boldsymbol{w}} \sim Q}[n\,\Delta(\tilde{\boldsymbol{w}})] \leq \mathrm{D}[Q \,\|\, P] + \log \mathrm{E}_{\tilde{\boldsymbol{w}} \sim P} \left[e^{n\,\Delta(\tilde{\boldsymbol{w}})} \right], \tag{3.11}$$

then we have that

$$\mathrm{E}_{\tilde{\boldsymbol{w}} \sim Q}[\Delta(\tilde{\boldsymbol{w}})] \leq \frac{\mathrm{D}[Q \,\|\, P] + \log K}{n}. \tag{3.12}$$

Note that the relative entropy $\mathrm{D}[Q \,\|\, P]$ between $Q(\tilde{\boldsymbol{w}})$ and $P(\tilde{\boldsymbol{w}})$ is identical to the relative entropy between $Q(\boldsymbol{w})$ and $P(\boldsymbol{w})$, because the factor $R(\boldsymbol{r})$ cancels out in the Radon-Nikodym derivative (recall Definition A.4).

We use the notion of convex (Legendre) duality (see Section A.3) to prove (3.11). $\mathrm{D}[Q \,\|\, P]$ is convex in Q, and its convex dual is given by the log partition

function $\log \mathrm{E}_P[\exp(\lambda(\tilde{\boldsymbol{w}}))]$ (see Equation A.8; $\boldsymbol{w}$ has to be replaced by $\tilde{\boldsymbol{w}}$), where the dual parameter $\lambda(\tilde{\boldsymbol{w}})$ is measurable w.r.t. $dP(\tilde{\boldsymbol{w}})$. To see this, we only have to consider $\lambda(\tilde{\boldsymbol{w}})$ such that $\mathrm{E}_P[\exp(\lambda(\tilde{\boldsymbol{w}}))] < \infty$. For such a candidate, define the Gibbs measure

$$dP_G(\tilde{\boldsymbol{w}}) = \frac{e^{\lambda(\tilde{\boldsymbol{w}})}}{\mathrm{E}_P\left[e^{\lambda(\tilde{\boldsymbol{w}})}\right]}\, dP(\tilde{\boldsymbol{w}}), \tag{3.13}$$

which is a probability measure relative to $P(\tilde{\boldsymbol{w}})$. The relative entropy is non-negative (see Section A.3), therefore

$$\begin{aligned} 0 \leq \mathrm{D}[Q \,\|\, P_G] &= \int \log \left(\frac{\mathrm{E}_P\left[e^{\lambda(\tilde{\boldsymbol{w}})}\right]}{e^{\lambda(\tilde{\boldsymbol{w}})}} \frac{dQ(\tilde{\boldsymbol{w}})}{dP(\tilde{\boldsymbol{w}})} \right) dQ(\tilde{\boldsymbol{w}}) \\ &= \mathrm{D}[Q \,\|\, P] + \log \mathrm{E}_P\left[e^{\lambda(\tilde{\boldsymbol{w}})}\right] - \mathrm{E}_Q[\lambda(\tilde{\boldsymbol{w}})]. \end{aligned} \tag{3.14}$$

Furthermore, if $\mathrm{D}[Q \,\|\, P]$ is finite, then the density dQ/dP exists, and the inequality becomes an equality for $\lambda(\tilde{\boldsymbol{w}}) = \log(dQ(\tilde{\boldsymbol{w}})/dP(\tilde{\boldsymbol{w}})) + c$ for any c. Now, equation (3.11) follows from (A.8) if we use $\lambda(\tilde{\boldsymbol{w}}) = n\,\Delta(\tilde{\boldsymbol{w}})$.

We can conclude the proof by noting the convexity of the relative entropy (see Section A.3) and using Jensen's inequality. Namely, if (3.12) holds for S, then

$$\begin{aligned} \mathrm{D}\left[\mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})] \,\|\, \mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[\boldsymbol{p}(\tilde{\boldsymbol{w}})]\right] &\leq \mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}\left[\mathrm{D}[\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}}) \,\|\, \boldsymbol{p}(\tilde{\boldsymbol{w}})]\right] \\ &\leq \frac{\mathrm{D}[Q \,\|\, P] + (L-1)\log(n+1) - \log\delta}{n}. \end{aligned} \tag{3.15}$$

If we compare this inequality with the inequality (3.9) for the "dumb" classifier based on P, we see that we have to pay a penalty $n^{-1}\mathrm{D}[Q \,\|\, P]$ for replacing the prior P by the posterior Q. Altogether, since $\hat{\boldsymbol{p}}(S,Q) = \mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})]$, $\boldsymbol{p}(Q) = \mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[\boldsymbol{p}(\tilde{\boldsymbol{w}})]$, we can combine (3.8) and the fact that for fixed S, (3.10) implies (3.15) in order to conclude that (3.5) must hold.

Theorem 3.1 is a special case of Theorem 3.2, and is obtained by setting $\mathcal{L} = \{0,1\}$ and $l(\boldsymbol{x}_*, y_*, \boldsymbol{w}, \boldsymbol{r}) = \mathrm{I}_{\{y(\boldsymbol{x}_*|\boldsymbol{w},\boldsymbol{r})\neq y_*\}}$. Then, it is easy to see that $\boldsymbol{p}(Q) = (1-\mathrm{gen}(Q), \mathrm{gen}(Q))$, $\hat{\boldsymbol{p}}(S,Q) = (1-\mathrm{emp}(S,Q), \mathrm{emp}(S,Q))$ and $\mathrm{D}[\hat{\boldsymbol{p}}(S,Q) \,\|\, \boldsymbol{p}(Q)] = \mathrm{D}_{\mathrm{Ber}}[\mathrm{emp}(S,Q) \,\|\, \mathrm{gen}(Q)]$. Theorem 3.1 follows from using (3.2).

3.2.3 Comments

We have seen above in Section 3.2.2 that the proof of the PAC-Bayesian theorem naturally decomposes into two parts. The first part is specific to the setting and

consists of proving a large deviation inequality of the style (3.8) for a "dumb" Gibbs classifier which selects its rule based on the prior P, independent of the sample S. The second part is generic and quantifies the slack in this inequality that we have to pay if we want to replace the "dumb" classifier based on P against the Gibbs rule of interested, based on the posterior Q. This slack is quantified directly by the relative entropy $\mathrm{D}[Q \,\|\, P]$ between posterior and prior. Thus, whenever our learning method holds it necessary to select a posterior distribution Q which is concentrated on a region deemed very unlikely under P, a high cost has to be paid in the bound.

The prior distribution P seems to enter the PAC-Bayesian theorem out of "thin air", yet its role is central as a parameter which can be tuned *a priori* to potentially tighten the bound.[5] The choice of P is of course constrained by the requirement of independence from the sample S. Given that, it should be chosen to give rise to small $\mathrm{D}[Q \,\|\, P]$ for samples S which we believe are likely to be observed for our task. If the PAC-Bayesian theorem is applied to an (approximate) Bayesian technique, this coincides with the typical Bayesian objective of coding available task prior knowledge into the prior distribution and the model class.

It is interesting to compare the terms in the PAC-Bayesian theorem with a frequently used Bayesian model selection criterion: the *log marginal likelihood*

$$\log P(\boldsymbol{y}) = \log \int P(\boldsymbol{y}|\boldsymbol{w})\, dP(\boldsymbol{w})$$

(see Sections A.6.2 and 2.1.3). If we employ the true posterior in the PAC-Bayesian theorem, i.e. $Q(\boldsymbol{w}) = P(\boldsymbol{w}|\boldsymbol{y})$, we have

$$\log P(\boldsymbol{y}) = \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\log \frac{P(\boldsymbol{y}|\boldsymbol{w})\, dP(\boldsymbol{w})}{dP(\boldsymbol{w}|\boldsymbol{y})}\right] = \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\log P(\boldsymbol{y}|\boldsymbol{w})\right] - \mathrm{D}[Q \,\|\, P], \quad (3.16)$$

therefore

$$-\frac{1}{n}\log P(\boldsymbol{y}) = \frac{\mathrm{D}[Q \,\|\, P]}{n} + \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{\boldsymbol{w}\sim Q}\left[-\log P(y_i|\boldsymbol{x}_i, \boldsymbol{w})\right].$$

Thus, the normalised negative log marginal likelihood which is minimised in Bayesian model selection, is the sum of the complexity term employed in the

[5] This can be seen as strong instance of the so-called *stratification technique*, although we average over a continuum of possible $\tilde{\boldsymbol{w}}$ and do not use the union bound (see Section 2.2.1).

PAC-Bayesian theorem and the sample average of $\mathrm{E}_{\boldsymbol{w}\sim Q}[-\log P(y|\boldsymbol{x},\boldsymbol{w})]$. The latter term, which we refer to as average likelihood, penalises training errors. Therefore, the log marginal likelihood incorporates a similar trade-off than the PAC-Bayesian theorem, using the same complexity term up to log factors. If Q is just an approximation to the posterior, we see from (2.16) that the r.h.s. of (3.16) is a lower bound on $\log P(\boldsymbol{y})$. For a very broad model class, the average likelihood will be small, while for a small model class, the posterior Gibbs rule will mis-classify more points in the training sample, leading to a larger average likelihood. However, for a large model class the prior $P(\boldsymbol{w})$ is necessarily rather broad and the derivative $dQ(\boldsymbol{w})/dP(\boldsymbol{w})$ will be large in the region where the posterior $Q(\boldsymbol{w})$ is concentrated, leading to a large complexity term $n^{-1}\mathrm{D}[Q\,\|\,P]$. Put simply, the density ratio between discriminants we consider to be sensible after and before having seen the data, should increase with growth of the model class.

We would like to stress that the *number* of parameters of a model class is not a sensible measure of complexity. This is fairly obvious, since we can take any model class and create a new one by adding a large number of parameters which do not or only slightly influence the rules. Introducing the notion of "effective number of parameters" helps only if this number is defined unambiguously and can be computed feasibly. For example, the non-parametric methods we consider in Section 3.3 can be seen as having an infinite number of parameters, however these parameters are regularised by the prior, and the conditioning on a finite amount of data will only render a finite number of these parameters any significant influence on predictions. The complexity measure $\mathrm{D}[Q\,\|\,P]$ behaves correctly in such situations, as the following argument suggests. Let $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2)$, and suppose that $\boldsymbol{w}_2$ has no influence on the rules, i.e. $P(y|\boldsymbol{x},\boldsymbol{w}) = P(y|\boldsymbol{x},\boldsymbol{w}_1)$. Then we have $dP(\boldsymbol{w}|\boldsymbol{y}) = dP(\boldsymbol{w}_1|\boldsymbol{y})dP(\boldsymbol{w}_2|\boldsymbol{w}_1)$ and $\mathrm{D}[P(\boldsymbol{w}|\boldsymbol{y})\,\|\,P(\boldsymbol{w})] = \mathrm{D}[P(\boldsymbol{w}_1|\boldsymbol{y})\,\|\,P(\boldsymbol{w}_1)]$, thus the complexity measure ignores $\boldsymbol{w}_2$ and its distribution.

We finally note that nowhere in the proof of Theorem 3.2 did we require a union bound (see Section 2.2.1), a distinctive advantage of the PAC-Bayesian

method. In a nutshell, while traditional techniques often perform a sort of "sphere-covering" of a given hypothesis space, then employ some (often loose) covering number arguments and the union bound, the PAC-Bayesian theorem employs expectations instead, first over the prior P, then changing P for the posterior Q at the "cost" of $n^{-1}\mathrm{D}[Q \,\|\, P]$. The slack comes from the fact that we use a linear lower bound to a convex function (see Section A.3).

3.2.4 The Case of General Bounded Loss

We stated and proved the PAC-Bayesian theorem above for the case of zero-one loss w.r.t. sets of size L. In this subsection, we provide a generalisation to to general bounded loss functions in Theorem 3.3 below. The proof, which is given in Appendix B.3, uses a "water-filling" argument due to [117].

We adopt the notation of Section 3.2.2, but now assume that there is a bounded loss function $l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_*, y_*)) \in [0,1]$ which quantifies the loss[6] one occurs when using the rule $\tilde{\boldsymbol{w}}$ in order to predict the target corresponding to $\boldsymbol{x}_*$ and the true target is y_*. For example, the zero-one loss for binary classification is given by $l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_*, y_*)) = \mathrm{I}_{\{y(\boldsymbol{x}_*|\tilde{\boldsymbol{w}}) \neq y_*\}}$. We use the notation $l(\tilde{\boldsymbol{w}}) = \mathrm{E}[l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_*, y_*))]$, where the expectation is over $(\boldsymbol{x}_*, y_*)$ drawn from the data distribution, and $\hat{l}(\tilde{\boldsymbol{w}}) = n^{-1} \sum_i l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_i, y_i))$. For fixed $\tilde{\boldsymbol{w}}$, $\hat{l}(\tilde{\boldsymbol{w}}) \to l(\tilde{\boldsymbol{w}})$ almost surely, and since l is bounded, the convergence rate is exponential in n. A typical large deviation inequality (see Section A.7) looks as follows. We have a nonnegative function ϕ on $[0,1]^2$ such that for every fixed $\tilde{\boldsymbol{w}}$:

$$\begin{aligned} \Pr\left\{\hat{l} \geq q\right\} &\leq e^{-n\phi(q,l)}, \quad q \geq l, \\ \Pr\left\{\hat{l} \leq q\right\} &\leq e^{-n\phi(q,l)}, \quad q \leq l, \end{aligned} \tag{3.17}$$

where $l = l(\tilde{\boldsymbol{w}})$, $\hat{l} = \hat{l}(\tilde{\boldsymbol{w}})$. We require that $\phi(q,l)$ is nondecreasing in $|q-l|$ and furthermore convex in (q,l), and $\phi(l,l) = 0$ for all $l \in [0,1]$. For simplicity, we also require that $\phi(\cdot, l)$ is differentiable on $(0,l)$ and $(l,1)$, for all $l \in (0,1)$. We will give examples for possible inequalities in a moment. Choose some $\delta \in (0,1)$.

[6] The notation $l(\cdot)$ used here should not be confused with the notation $l \in \mathcal{L}$ used in subsection 3.2.2.

Theorem 3.3 (PAC-Bayesian Theorem, Bounded Loss Functions) *For any data distribution over $\mathcal{X}\times\{-1,+1\}$, we have that the following bound holds, where the probability is over random i.i.d. samples $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1,\ldots,n\}$ of size n drawn from the data distribution:*

$$\Pr_S \left\{ \begin{array}{c} \phi\left(\mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[\hat{l}(\tilde{\boldsymbol{w}})], \mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[l(\tilde{\boldsymbol{w}})]\right) > \frac{1}{n-1}\left(\mathrm{D}[Q \,\|\, P] + \log\frac{2n+1}{\delta}\right) \\ \text{for some } Q \end{array} \right\} \leq \delta. \quad (3.18)$$

The proof is given in Appendix B.3. Note that the function $\phi(\mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[\hat{l}(\tilde{\boldsymbol{w}})], \cdot)$ can be inverted just as easily as $\mathrm{D}_{\mathrm{Ber}}$ above (see equation (3.1)) in order to obtain upper and lower bounds on $\mathrm{E}_{\tilde{\boldsymbol{w}}\sim Q}[l(\tilde{\boldsymbol{w}})]$.

If we instantiate this theorem for zero-one loss using Chernoff's bound, we essentially obtain Theorem 3.1. In fact, Chernoff's bound holds just as well for general bounded loss and has the form of (3.17) with $\phi(q,l) = \mathrm{D}_{\mathrm{Ber}}[q \,\|\, l]$ (Theorem A.3). The original formulation in [117] uses the Hoeffding bound (see Appendix A.7) which is significantly less tight than the Chernoff bound if the expected empirical error is far from 1/2. Other more specialised large-deviation inequalities can be used instead, and even if they do not come in the form of (3.17), the proof can probably be adapted. However, note that the constraint on $\phi(q,l)$ to be convex in (q,l) is rather crucial for the last step of the proof. More specifically, we have to upper bound $\phi(\mathrm{E}_Q[\hat{l}(\tilde{\boldsymbol{w}})], \mathrm{E}_Q[l(\tilde{\boldsymbol{w}})])$ in terms of the bound on $\mathrm{E}_Q[\phi(\hat{l}(\tilde{\boldsymbol{w}}), l(\tilde{\boldsymbol{w}}))]$, which is trivial if ϕ is convex.

3.2.5 An Extension to the Bayes Classifier

In practice, when comparing Gibbs and Bayes classifier for the same posterior distribution $Q(\boldsymbol{w})$ directly, it turns out that the Bayes variant often performs better than the Gibbs variant (the latter is hardly used in practice for this reason). In this section, we will have a closer look on their relationship and suggest a simple extension of the PAC-Bayesian theorem for binary Bayes classifiers.

We have introduced Gibbs, Bayes and Bayes voting classifiers in Section 3.1.2. Recall that throughout this thesis we assume that the noise model is symmetric in the sense that it is a function of yu: $P(-y|u) = P(y|-u)$. For simplicity, we

assume in this section that if a bias parameter b is used, it has already been added to u. In general, the Bayes, Bayes voting and predictive classifier are all different, but we will be interested exclusively in the case that the distribution of $u(\boldsymbol{x} \mid \boldsymbol{w})$ induced by Q is symmetric around its mean for every $\boldsymbol{x}$. In this case, all Bayes classifier variants agree. In fact, it is easy to see that for every nondecreasing $f(u)$ with $f(-u) + f(u) = \tau$, the classifiers $\operatorname{sgn}(\mathrm{E}_Q[f(u(\boldsymbol{x}_* \mid \boldsymbol{w}))] - \tau/2)$ are identical. Note that if this assumption is violated, the concepts of Gibbs, Bayes and Bayes voting classifiers become questionable anyway and the predictive classifier should be used. Fortunately, all (approximate) Bayesian methods discussed in this thesis fulfil this assumption. Define the errors

$$e_{\mathrm{Gibbs}}(\boldsymbol{x}_*, y_*) = \mathrm{E}_Q[\mathrm{I}_{\{\operatorname{sgn} u(\boldsymbol{x}_* \mid \boldsymbol{w}) \neq y_*\}}],$$
$$e_{\mathrm{Bayes}}(\boldsymbol{x}_*, y_*) = \mathrm{I}_{\{\operatorname{sgn} \mathrm{E}_Q[u(\boldsymbol{x}_* \mid \boldsymbol{w})] \neq y_*\}}.$$

Furthermore, for $A \in \{\mathrm{Gibbs}, \mathrm{Bayes}\}$, define $e_A = \mathrm{E}_{(\boldsymbol{x}_*, y_*)}[e_A(\boldsymbol{x}_*, y_*)]$ where the expectation is over the data distribution. Intuitively, the Bayes classifier should outperform the Gibbs variant if the trained model represents the data distribution well. Assume for now that the data distribution *is* identical to the model-generative one: for given $\boldsymbol{x}_*$, $y_* \sim P(y_* \mid u(\boldsymbol{x}_* \mid \boldsymbol{w}))$, $\boldsymbol{w} \sim Q$. Condition on $\boldsymbol{x}_*$ and write $u = u(\boldsymbol{x}_* \mid \boldsymbol{w}) = \bar{u} + v$, where $\bar{u} = \mathrm{E}_Q[u]$ and v has an even density function. For simplicity, we write $u \sim Q$, $v \sim Q$ meaning the corresponding distributions induced by Q over $\boldsymbol{w}$ (for fixed $\boldsymbol{x}_*$). Consider sampling $u_1, u_2 \sim Q$ independently, furthermore $y_2 \sim P(y_2|u_2)$. Both Gibbs and Bayes classifier err if $\operatorname{sgn} u_1 = \operatorname{sgn} \bar{u}$, $y_2 \neq \operatorname{sgn} u_1$, but if $\operatorname{sgn} u_1 \neq \operatorname{sgn} \bar{u}$, they differ depending on the relationship between y_2 and $\operatorname{sgn} \bar{u}$. If $E = \{\operatorname{sgn} u_1 \neq \operatorname{sgn} \bar{u}\}$, then

$$e_{\mathrm{Gibbs}} - e_{\mathrm{Bayes}} = \mathrm{E}_{\boldsymbol{x}_*} \mathrm{E}\left[\Pr\{E\} \left(\Pr\{y_2 = \operatorname{sgn} \bar{u}\} - \Pr\{y_2 \neq \operatorname{sgn} \bar{u}\} \right) \,\middle|\, \boldsymbol{x}_* \right].$$

$\Pr\{E\}$ is the probability of the tail $\{v \geq |\bar{u}|\}$ under Q. Note that we have used that the event $\{y_2 = \operatorname{sgn} \bar{u}\}$ is independent of E, given $\boldsymbol{x}_*$. The conditional difference of the errors is positive if $\operatorname{sgn} \bar{u} \neq 0$, showing that the Bayes classifier outperforms the Gibbs variant in this situation. On the other hand, it is easy to construct a "malicious" setting in which the Gibbs classifier does better than the

Bayes variant (but see below), but recall from Section 3.1.1 that the relevance of such arguments may be limited in practice.

If nothing is known about the data distribution, we can relate e_{Bayes} and e_{Gibbs} in a coarser sense, by noting that if $e_{\text{Bayes}}(\boldsymbol{x}_*, y_*) = 1$, then $e_{\text{Gibbs}}(\boldsymbol{x}_*, y_*) \geq 1/2$, thus $e_{\text{Bayes}} \leq 2\, e_{\text{Gibbs}}$ (as remarked in [72], Lemma 5.3).[7] Therefore, Theorem 3.1 applies to the Bayes classifiers as well. Although we obtained this extension without efforts, it is not really what we are ideally looking for. First, the bound on the Bayes classifier error is certainly over-pessimistic.[8] Second, the generalisation error bound on the Bayes classifier is in terms of the expected empirical error of the *Gibbs* classifier: even if we prefer the Bayes classifier in practice, we have to evaluate its Gibbs variant in order to obtain performance guarantees. Third, the argument does not carry through to more than two classes. Very recently, Meir and Zhang [121] obtained a PAC-Bayesian margin bound for the Bayes voting classifier, combining a new inequality based on Rademacher complexities with the convex duality step. We discuss this result in Section 3.4.1. However, the Meir/Zhang result can be criticised on other grounds (see Section 3.4.1.1) and did not render non-trivial guarantees in our experiments (see Section 3.5.6), while the Gibbs theorem certainly did. In light of practical evidence, the aim is to provide a PAC-Bayesian theorem for Bayes-type classifiers which is at least as tight as the Gibbs theorem in practically relevant situations, and at least to our knowledge this remains an open problem.

3.2.6 Some Speculative Extensions

Here, at the end of Section 3.2 we take the liberty of collecting some more or less speculative thoughts which are not driven to a definite conclusion. In Section 3.2.6.1, we note that in principle the complexity measure $n^{-1}\mathrm{D}[Q \,\|\, P]$ could be replaced by other such measures, as long as they are convex in Q for every P. In Section 3.2.6.2 we show that the essential slack in the PAC-Bayesian bound

[7] Even without any assumption on the distributions of $u(\boldsymbol{x}_* \,|\, \boldsymbol{w})$, this is true for *Bayes voting* and Gibbs classifier.

[8] One can construct situations to show that it is tight in principle, but recall from Section 3.1.1 that such "tightness" arguments may be misleading in practice.

can be written down explicitly. Gaining knowledge about the behaviour of slack, especially about its degree of dependence on hyperparameters may be a key issue if PAC bounds are to be used for model selection in practice.

3.2.6.1 Generalisation to Other Complexity Measures

The nature of the proof given in Section 3.2.2 allows us to think about further generalisations. Recall that the key for transforming the uninteresting bound (3.9) into the statement of the PAC-Bayesian theorem is the characterisation (A.8) of the relative entropy $\mathrm{D}[Q \,\|\, P]$ in terms of its convex dual (see Section A.3). Suppose that $d(P, Q)$ is any divergence measure between P and Q which is convex in Q for all P. Then, there exists a convex dual $g(P, \lambda)$ such that

$$d(P, Q) = \max_{\lambda(\tilde{\boldsymbol{w}})} \left(\mathrm{E}_{\tilde{\boldsymbol{w}} \sim Q} \left[\lambda(\tilde{\boldsymbol{w}}) \right] - g(P, \lambda) \right),$$

and $g(P, \lambda)$ is itself convex in λ for every P. For a general divergence measure $d(P, Q)$, the following program may be feasible to entertain. First, make sure that $d(P, Q)$ is convex in Q for every P. Second, determine its convex dual $g(P, \lambda)$. Third, prove a large deviation bound of the form

$$\Pr_S \left\{ e^{g(P, n\,\Delta(\tilde{\boldsymbol{w}}))} > \frac{K}{\delta} \right\} \leq \delta, \tag{3.19}$$

where K is polynomial in n. This would serve as equivalent for (3.9), and the generic part of the proof could be followed in order to obtain a PAC-Bayesian theorem which asserts that

$$\mathrm{D}\left[\mathrm{E}_{\tilde{\boldsymbol{w}} \sim Q}[\hat{\boldsymbol{p}}(\tilde{\boldsymbol{w}})] \,\|\, \mathrm{E}_{\tilde{\boldsymbol{w}} \sim Q}[\boldsymbol{p}(\tilde{\boldsymbol{w}})] \right] \leq \frac{d(P, Q) + \log K - \log \delta}{n}$$

with probability at least $1 - \delta$ over draws of S. The feasibility of this approach depends on several factors. Of course, $d(P, Q)$ must be convex in Q and itself feasible to compute. Then, we have to determine the convex dual $g(P, \lambda)$ in closed form, or at least $g(P, n\,\Delta)$. Second (and probably most difficult) we have to prove the bound (3.19) for our prior at hand. Note that we are not necessarily constrained to use the particular divergence $\Delta(\tilde{\boldsymbol{w}})$ as defined in subsection 3.2.2: the latter was chosen for convenience w.r.t. proving (3.9). However, $\Delta(\tilde{\boldsymbol{w}})$ has to

be convex in $\tilde{\boldsymbol{w}}$ for every sample S. We have not yet pursued this program for any divergence $d(P, Q)$ other than the relative entropy.

3.2.6.2 The Slack Term in the PAC-Bayesian Theorem

Several authors have suggested to minimise PAC upper bounds in order to do model selection. In Section 2.2.4, we argue that this is theoretically justified only if one can prove that the *slack* in the bound, i.e. the difference between bound value and true generalisation error, is significantly less variable w.r.t. hyperparameters to be selected (over a range of interest) than the generalisation error itself. It is not even straightforward to formalise this requirement in the strict PAC sense, and proving it will most probably be much harder than the bound itself. Nevertheless, the phenomenon occurs in practice on non-trivial real world examples (see Section 3.5.5), and it would be very valuable to obtain some analytical results in order to understand it at least on toy models.

An interesting consequence of the simple proof of the PAC-Bayesian Theorem 3.1 given in Section 3.2.2 is that we can essentially write down the slack analytically. For simplicity, we deal with deterministic rules only, i.e. $\tilde{\boldsymbol{w}} = \boldsymbol{w}$. Inspecting the proof, we see that there are three sources of slack: the initial arguments leading to (3.8), the plugging in of $n\Delta(\boldsymbol{w})$ for $\lambda(\boldsymbol{w})$ in (A.8) and the final application of Jensen's inequality in (3.15). The bound in (3.8) leads to a typically minor contribution to the PAC-Bayesian gap bound, and the final application of Jensen's inequality should be fairly tight if the posterior Q is rather concentrated,[9] leaving us with the typically dominating slack coming from the "wrong" choice for $\lambda(\boldsymbol{w})$ in (A.8). From (3.14) we see that this slack in the right hand side $\varepsilon(\delta, S, P, Q)$ in (3.5) is $\mathrm{D}[Q \,\|\, P_G]$, where P_G is the Gibbs measure defined in (3.13). Suppose that $Q(\boldsymbol{w})$ has the form

$$dQ(\boldsymbol{w}) = Z^{-1} e^{\sum_{i=1}^{n} \phi_i(\boldsymbol{w})} dP(\boldsymbol{w}), \quad Z = \mathrm{E}_{\boldsymbol{w} \sim P}\left[e^{\sum_{i=1}^{n} \phi_i(\boldsymbol{w})} \right],$$

where $\phi_i(\boldsymbol{w}) = \phi(\boldsymbol{x}_i, y_i; \boldsymbol{w})$. For example, the true Bayesian posterior is obtained

[9] This could be tested using random sampling for a particular architecture.

if $\phi(\boldsymbol{x}_i, y_i; \boldsymbol{w}) = \log P(y_i|\boldsymbol{x}_i, \boldsymbol{w})$. Then,

$$\begin{aligned}\mathrm{D}[Q \,\|\, P_G] &= \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\log \frac{dQ(\boldsymbol{w})}{dP_G(\boldsymbol{w})}\right] \\ &= \sum_{i=1}^{n} \mathrm{E}_{\boldsymbol{w}\sim Q}\left[\phi_i(\boldsymbol{w}) - \Delta(\boldsymbol{w})\right] - \log \frac{\mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{\sum_i \phi_i(\boldsymbol{w})}\right]}{\mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\Delta(\boldsymbol{w})}\right]}.\end{aligned}$$

We can also write

$$\mathrm{D}[Q \,\|\, P_G] = n\,\mathrm{E}_{\boldsymbol{w}\sim Q}\left[\Delta(\boldsymbol{w})\right] - \log \mathrm{E}_{\boldsymbol{w}\sim P}\left[e^{n\Delta(\boldsymbol{w})}\right] - \mathrm{H}[Q(\boldsymbol{w})].$$

Admittedly, these expressions are not very useful per se, expect for showing that the slack will be small if $\Delta(\boldsymbol{w})$ is close to most of the $\phi_i(\boldsymbol{w})$.

3.3 Application to Gaussian Process Classification

In this section, we apply the PAC-Bayesian Theorem 3.1 to a wide class of Gaussian process models for binary classification. The class is defined in Section 2.1.3, and in Section 3.3.1 we show how the bound terms can be computed for any member. In Sections 3.3.2, 3.3.3 and 3.3.4, we give specific examples for some well-known GP approximations (see Section 2.1.3).

3.3.1 PAC-Bayesian Theorem for GP Classification

In this section, we specialise the PAC-Bayesian Theorem 3.1 to Gaussian process models for binary classification, incorporating a wide class of approximate inference methods. The GP binary classification model has been introduced in Section 2.1.2 (both the logit and probit noise model discussed there satisfy the symmetry condition required here). Recall that $\boldsymbol{K} \in \mathbb{R}^{n,n}$ denotes the covariance matrix evaluated over the training input points $\{\boldsymbol{x}_i\}$, and $\boldsymbol{u} = (u_i)_i \in \mathbb{R}^n$ are the latent outputs at these points. This non-parametric model can be seen as special case of the scenario of Section 3.1.2 with $\boldsymbol{w} \equiv u(\cdot)$, i.e. the "weights" are the complete latent process, and $u(\boldsymbol{x}|\boldsymbol{w}) \equiv u(\boldsymbol{x})$. Alternatively, we could develop the process $u(\boldsymbol{x})$ in an eigensystem of the covariance kernel K and parameterise

it in terms of a countable number of weights (the "weight space view" of Section 2.1.1), however the presentation in the "process view" turns out to be much simpler.

The general class of approximation methods we are interested in here is defined in Section 2.1.3. The posterior $P(\boldsymbol{u}|S)$ is approximated by a Gaussian $Q(\boldsymbol{u}|S)$ of the general form (2.12). For most schemes, the covariance matrix $\boldsymbol{A}$ of $Q(\boldsymbol{u}|S)$ is further restricted to the form (2.13), thus the approximation is defined in terms of $\boldsymbol{K}$ and further $O(d)$ parameters, $d \leq n$. Then, the approximate predictive distribution is Gaussian with mean and variance given in (2.14). In order to apply the PAC-Bayesian theorem to this case, we only have to show how to compute the terms defining the gap bound value: the expected empirical error and the relative entropy $\mathrm{D}[Q \,\|\, P]$. The former is just the empirical average over

$$e_{\mathrm{Gibbs}}(\boldsymbol{x}_*, y_*) = \mathrm{Pr}_{u_* \sim Q(u_* \,|\, \boldsymbol{x}_*, S)}\{\mathrm{sgn}(u_* + b) \neq y_*\} = \Phi\left(\frac{-y_*(\mu(\boldsymbol{x}_*) + b)}{\sigma(\boldsymbol{x}_*)}\right), \tag{3.20}$$

where $\Phi(\cdot)$ denotes the c.d.f. of $N(0,1)$. The relative entropy $\mathrm{D}[Q \,\|\, P]$ has been determined in (2.11). Using (A.17), we obtain

$$\mathrm{D}[Q \,\|\, P] = \frac{1}{2}\left(\log\left|\boldsymbol{A}^{-1}\boldsymbol{K}\right| + \mathrm{tr}\left(\boldsymbol{A}^{-1}\boldsymbol{K}\right)^{-1} + \boldsymbol{\xi}^T\boldsymbol{K}_I\boldsymbol{\xi} - n\right). \tag{3.21}$$

If $\boldsymbol{A}$ is of the special form (2.13), this simplifies to

$$\mathrm{D}[Q \,\|\, P] = \frac{1}{2}\left(\log|\boldsymbol{B}| + \mathrm{tr}\,\boldsymbol{B}^{-1} + \boldsymbol{\xi}^T\boldsymbol{K}_I\boldsymbol{\xi} - d\right), \tag{3.22}$$

where $\boldsymbol{B}$ is defined in (2.14). These formulae depend on the generic parameters $\boldsymbol{\xi}$ and $\boldsymbol{A}$ (or I and $\boldsymbol{D}$) of the posterior approximation $Q(\boldsymbol{u}|S)$. In Sections 3.3.2 and 3.3.3, we show how to compute the relative entropy for a range of concrete GPC approximations. A simple way to compute (3.22) is to use the Cholesky decomposition $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T$ (see Appendix A.2.2). Then, $\log|\boldsymbol{B}| = 2\log|\,\mathrm{diag}^2\,\boldsymbol{L}|$, and $\mathrm{tr}\,\boldsymbol{B}^{-1}$ can be computed from $\boldsymbol{L}$ using the same number of operations as required for $\boldsymbol{B} \to \boldsymbol{L}$.

We can now simply plug in (3.21) or (3.22) and (3.20) into the terms of Theorem 3.1 in order to obtain a PAC-Bayesian theorem for GP binary Gibbs classifiers. It is important to note that for this theorem to be valid, the prior

P and the model have to be fixed *a priori*, i.e. free hyperparameters of the covariance function K or the noise model $P(y|u)$ have to be chosen *independently* of the training sample S. For example, the widely used practice of choosing such parameters by maximising an approximation to the log marginal likelihood $\log P(S)$ (see Sections 2.1.3 and A.6.2) is *not* compatible with using the theorem for a statistical test. It is easy to modify Theorem 3.1 to allow for model selection amongst a finite set (size polynomial in n) of hyperparameter candidates, by a straightforward application of the union bound (see Section 2.2.1). This introduces a $O(\log n)$ term in the numerator of $\varepsilon(\delta, n, P, Q)$ in (3.3). However, choosing the hyperparameters by continuous optimisation is not admissible.

Note also that Theorem 3.2 in principle applies to multi-class GP methods (see Section 2.1.2). As mentioned there, the implementation of such methods is, however, quite involved, and for most methods, additional approximations are required to avoid a scaling quadratic in the number of classes. In terms of Theorem 3.2, the computation of $\hat{\boldsymbol{p}}(S, Q)$ may not be analytically tractable, and special approximations may have to be considered in order not to compromise the bound statement. This program is subject to future work.

3.3.2 Laplace Gaussian Process Classification

Let us specialise the generic framework of the previous section to Laplace GP binary classification [210], discussed briefly in Section 2.1.3. We end up with a Gaussian posterior approximation with mean $\boldsymbol{\xi}$ and covariance $\boldsymbol{A} = (\boldsymbol{K}^{-1} + \boldsymbol{D})^{-1}$, where $\boldsymbol{\xi}$, $\boldsymbol{D}$ depend on the posterior mode $\hat{\boldsymbol{u}}$ via (2.15). Note that in order to evaluate the Gibbs classifier, the predictive variance $\sigma^2(\boldsymbol{x}_*)$ has to be computed, while the (approximate) Bayes classifier is $\mathrm{sgn}(\mu(\boldsymbol{x}_*) + b)$, independent of $\sigma^2(\boldsymbol{x}_*)$ (see Equation 2.14).[10] The downside of this is that the Gibbs classifier requires $O(n^2)$ for each evaluation, while the Bayes classifier is $O(n)$ if an uncertainty estimate is not required. We show in Appendix B.4 how to evaluate the Gibbs classi-

[10] Recall from Section 3.1.2 that since $Q(u_*|\boldsymbol{x}_*, S)$ is symmetric around $\mu(\boldsymbol{x}_*)$, Bayes, Bayes voting and predictive classifiers predict the same target. The independence of these classifiers of $\sigma^2(\boldsymbol{x}_*)$ could be interpreted as weakness of the Gaussian approximation, since the true posterior for u_* can be skew. Error bars for the classifiers *do* of course depend on the predictive variance.

fier more efficiently using sparse approximations together with rejection sampling techniques, resulting in $O(n)$ (average case) per prediction.

Experimental results are given in Section 3.5. We can gain some insight into the gap bound value by analysing the relative entropy term $\mathrm{D}[Q \,\|\, P]$ (see Equation 3.22) in Theorem 3.1, as applied to Laplace GPC. In normal situations (i.e. δ not extremely small), the expression $\varepsilon(\delta, n, P, Q)$ in (3.3) is dominated by this term.

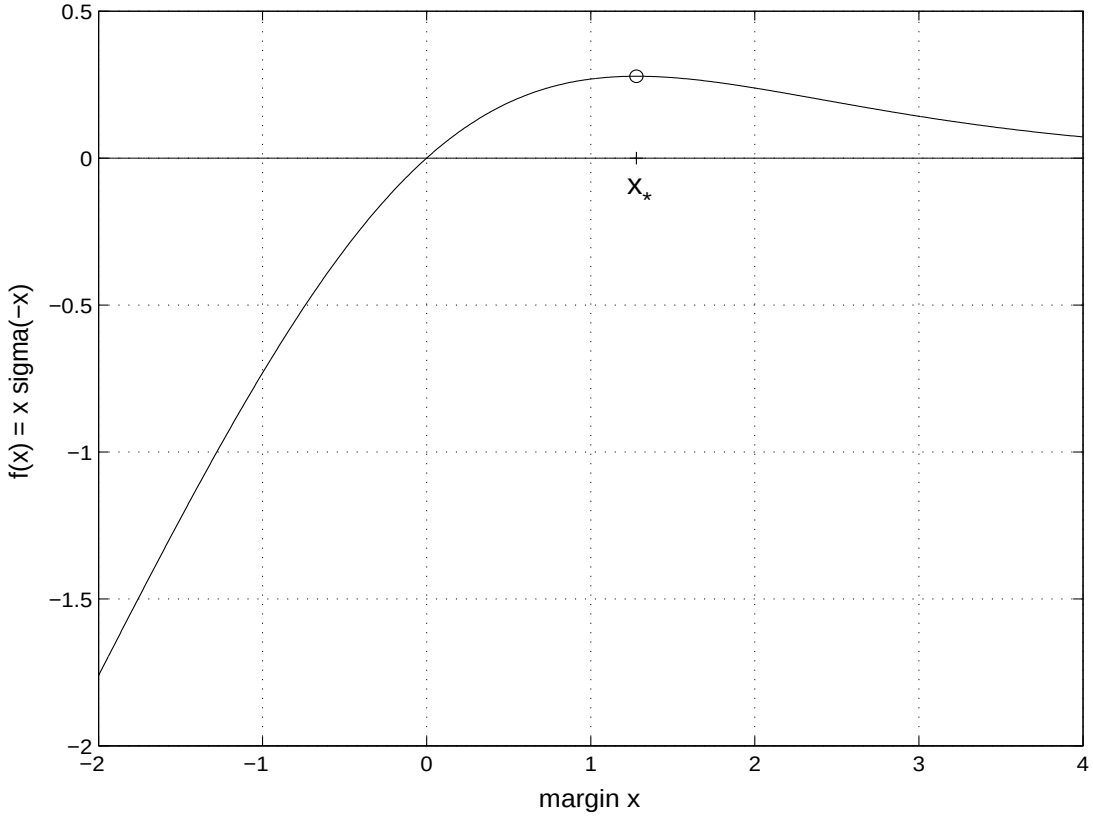

Figure 3.1: Relation between margin and gap bound part

We assume that $P(y|u)$ is logit noise. First, use (2.15) to see that $\boldsymbol{\xi}^T \boldsymbol{K} \boldsymbol{\xi} = \hat{\boldsymbol{u}}^T \boldsymbol{\xi} = \sum_i y_i \hat{u}_i \sigma(-y_i \hat{u}_i) = \sum_i f(y_i \hat{u}_i)$, where $f(x) = x\,\sigma(-x)$ and $y_i \hat{u}_i$ is the margin[11] at example $(\boldsymbol{x}_i, y_i)$. $f(x)$ is plotted in Figure 3.1. It is maximal at $x_* \approx 1.28$, converges to 0 exponentially quickly for $x \to \infty$ and behaves like $x \mapsto x$ for

[11]The margin is a learning-theoretical concept frequently used in the context of PAC bounds for averaging (Bayes) classifiers (see for example Section 3.4.1).

$x \to -\infty$. Thus, at least w.r.t. the third term in (3.22), classification mistakes (i.e. $y_i \hat{u}_i < 0$) render a negative contribution to the gap bound value. This is what we expect for a Bayesian architecture. Namely, the method choses a *simple* solution, at the expense of making this mistake, yet with the goal to prevent disastrous over-fitting. In our bound, we are penalised by a higher expected empirical error, but we should be rewarded by a smaller gap bound term. Now to the remaining terms in (3.22). The matrix $\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{D}^{1/2}\boldsymbol{K}\boldsymbol{D}^{1/2}$ is positive definite, and all its eigenvalues are ≥ 1. Furthermore, by taking any unit vector $\boldsymbol{z}$ and using the Fisher-Courant min-max characterisation of the eigenvalue spectrum of Hermitian matrices (e.g., [78], Sect. 4.2), we have $\boldsymbol{z}^T\boldsymbol{B}\boldsymbol{z} = 1 + (\boldsymbol{D}^{1/2}\boldsymbol{z})^T\boldsymbol{K}(\boldsymbol{D}^{1/2}\boldsymbol{z}) \leq 1 + \lambda_{\max}\boldsymbol{z}^T\boldsymbol{D}\boldsymbol{z} < 1 + \lambda_{\max}/4$, where $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{K}$. Here, (2.15) implies that the coefficients of $\operatorname{diag}\boldsymbol{D}$ are all $< 1/4$. Thus, all eigenvalues of $\boldsymbol{B}$ lie in $(1, 1 + \lambda_{\max}/4)$. By analysing $(1/2)\log|\boldsymbol{B}| + (1/2)\operatorname{tr}\boldsymbol{B}^{-1}$ for general[12] $\boldsymbol{B} \succeq \boldsymbol{I}$, using a spectral decomposition of $\boldsymbol{B}$, we see that this term must lie between $n/2$ and $(n/2)(\log(1+\lambda_{\max}/4) + (1+\lambda_{\max}/4)^{-1})$, although these bounds are not very tight.

3.3.3 Sparse Greedy Gaussian Process Classification

In practice, the applicability of Gaussian process or other kernel methods is often severely restricted by their unfortunate scaling: $O(n^3)$ time for first-level inference, $O(n^2)$ space. Sparse approximations to GP models can improve vastly on these figures and are therefore of considerable practical importance. In Chapter 4, we discuss sparse approximations in detail and present a range of different schemes for approximate inference. All of these lie within the class described in Section 2.1.3, so that Theorem 3.1 applies to their Gibbs classifier versions. In this section, we focus on the simple IVM scheme introduced in Section 4.4.1, the most efficient of the algorithms discussed in this thesis.

The parametric representation for IVM is described in Section 4.4.1 and Appendix C.3.1. In terms of the placeholders of Section 2.1.3 we have $\boldsymbol{D} = \boldsymbol{\Pi}_I$,

[12] $\boldsymbol{B} \succeq \boldsymbol{I}$ means that $\boldsymbol{B} - \boldsymbol{I}$ is positive semidefinite. See [24] for details about such generalised inequalities.

where I is the active set, furthermore $\boldsymbol{\xi} = \boldsymbol{I}_{\cdot,I}\boldsymbol{\Pi}_I^{1/2}\boldsymbol{L}^{-T}\boldsymbol{\beta}$. The relative entropy term is given by

$$\mathrm{D}[Q \,\|\, P] = \frac{1}{2}\left(\log|\boldsymbol{B}| + \operatorname{tr}\boldsymbol{B}^{-1} + \|\boldsymbol{\beta}\|^2 - \left\|\boldsymbol{L}^{-T}\boldsymbol{\beta}\right\|^2 - d\right).$$

The predictive distribution $Q(u_*|\boldsymbol{x}_*, S)$ is derived in Section 4.4.1. An evaluation of the Gibbs classifier costs $O(d^2)$ (one back-substitution with $\boldsymbol{L}$), while the computation of $\mathrm{D}[Q \,\|\, P]$ costs $O(d^3)$. The expected empirical error can be computed in $O(n\,d^2)$.

As discussed in Section 4.3, the non-sparse equivalent of the IVM (i.e. $I = \{1, \ldots, n\}$) is the cavity TAP method of [139], and Theorem 3.1 can be applied to this approximation using the formulae given here. Also note that although the IVM is a compression scheme (in the sense defined in Appendix B.6), the PAC-Bayesian theorem is *not* a compression bound: it holds for sparse non-compressing methods (such as PLV, see Section 4.4.2) just as well. We will see in Section 3.5.4 that the PAC-Bayesian theorem can be significantly tighter in practice than a standard PAC compression bound when applied to the same compression scheme.

3.3.4 Minimum Relative Entropy Discrimination

The MRED framework as probabilistic interpretation of large margin (discrimination) methods such as SVM has been introduced in [83] and is discussed in Section 2.1.6. The formulation allows a direct application of the PAC-Bayesian theorem 3.1 to Gibbs version of the corresponding classifiers.

Recall that the prior process P and the MRED "posterior" process Q have the same covariance, thus

$$\mathrm{D}[Q \,\|\, P] = \frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{K}\boldsymbol{\xi} = \frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{Y}\boldsymbol{K}\boldsymbol{Y}\boldsymbol{\lambda}.$$

Therefore, the application of Theorem 3.1 to MRED is straightforward. However, note that the Gibbs variant of the MRED classifier is typically not a very useful method in practice, due to the failure of MRED (and large margin methods in general) to provide non-trivial estimates of predictive variance. The predictive

variance of the latent variable u_* is the same as the corresponding prior variance which can be large, thus the Gibbs variant is likely to perform much worse than the Bayes (averaging) one for points close to a decision boundary even if they lie close to training points.

3.4 Related Work

In this section, we mention some closely related work. Our focus is on a result very recently proved in [121], giving a PAC-Bayesian theorem for Bayes voting classifiers, as opposed to Theorem 3.1 which holds for Gibbs classifiers. In Section 3.4.1.2, we show a possible route towards extending this result to the case of GP regression.

The literature on PAC bounds for kernel methods is large and will not be reviewed here (the reader may consult [72, 161, 197]). We provide a brief introduction to PAC bounds in Section 2.2. Shawe-Taylor and Williamson [176] present a PAC analysis of a Bayesian estimator. The notion of PAC-Bayesian theorems has been developed by McAllester [116, 115, 117] who also gave a proof of a slightly less tight version of Theorem 3.1. The theorems in [116] can be seen as special case for a restricted class of "cut-off posteriors" (where the log likelihood is an indicator function). McAllester's theorems have been used in a number of later papers. Herbrich and Graepel [73] use the theorems in [116] to obtain a margin bound on SVM. Seeger, Langford, and Megiddo [172] combine the "sampling" approach from [159] with the PAC-Bayesian theorem to obtain a Bayes classifier version. Langford and Caruana [95] apply McAllester's theorem to multi-layer perceptrons. Langford and Shawe-Taylor [94] provide another extension of Theorem 3.1 to Bayes classifiers, however with the same drawbacks as the one in Section 3.2.5. The interesting feature of this work is that it shows how to apply the PAC-Bayesian theorem to averaging (Bayes) classifiers which do not provide estimates of predictive variances. A similar idea has been proposed in [73], however with an unfortunate dependence on the dimensionality of the weight space. There is a considerable literature on PAC bounds for combined (mixture)

classifiers and multi-layered classifiers. Recent results are given by Koltchinskii and Panchenko [91] where other references can be found, see also [159].

3.4.1 The Theorem of Meir and Zhang

The PAC-Bayesian Theorem 3.1 holds for Gibbs classifiers only, while in practice Bayes or Bayes voting classifiers are used much more frequently, due to their typically better performance. Recently, Meir and Zhang [121] presented a PAC-Bayesian margin bound which applies to Bayes voting classifiers (recall that for the approximations we are interested in, Bayes and Bayes voting classifiers are identical, see Section 3.1.2). They obtain their result by combining a recent bound proved in [91] with a convex duality step identical to the second part of our proof. In this section, we present their result and compare it with the PAC-Bayesian Theorem 3.1. In fact, we derive a slightly different theorem which is closer to the relevant Theorem 2 in [91].

Meir and Zhang [121] consider functions $f(\boldsymbol{x}|Q) = \mathrm{E}_{\boldsymbol{w}\sim Q}[y(\boldsymbol{x}|\boldsymbol{w})]$, $y(\boldsymbol{x}|\boldsymbol{w}) = \mathrm{sgn}(u(\boldsymbol{x}|\boldsymbol{w}) + b)$. Allowing for the choice of a prior P, they define

$$\mathcal{Q}_A = \left\{ Q \,\middle|\, \mathrm{D}[Q \,\|\, P] \leq A \right\}, \; \mathcal{F}_A = \left\{ f(\boldsymbol{x}|Q) \,\middle|\, Q \in \mathcal{Q}_A \right\}.$$

Note that Meir and Zhang claim that their theorem holds for arbitrary classes $\mathcal{F}_A$, while based on the theorems in [91], we were only able to reproduce their result under the additional condition[13] that all $\mathcal{F}_A$ are closed under multiplication with -1. In other words, we require that for every $A > 0$ and every $f \in \mathcal{F}_A : \; -f \in \mathcal{F}_A$. Similar to structural risk minimisation bounds [197], they use a nested hierarchy $\{\mathcal{F}_{A_j}\}$ defined *a priori* via an unbounded sequence $A_1 < A_2 < \ldots$ together with a distribution $(p_j)_j$ over $\mathbb{N}$. Finally, let $\phi(x)$ be the function which is equal to $1 - x$ in $[0, 1]$, constant 1 for $x \leq 0$ and constant 0 for $x \geq 1$. Now, choose a $\delta \in (0, 1)$ and a $c > 1$.

Theorem 3.4 ([121]) *For any data distribution over $\mathcal{X} \times \{-1, +1\}$ we have that the following bound holds, where the probability is over random i.i.d. samples*

[13] The problem will be removed in a longer version of their paper (Tong Zhang, personal communication).

$S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\}$ of size n drawn from the data distribution:

$$\Pr_S \Bigg\{ \Pr_{(\boldsymbol{x}_*, y_*)} \{ y_* f(\boldsymbol{x}_* | Q) \leq 0 \} > \inf_{\kappa \in (0,1]} B(\kappa, S, Q) + \sqrt{\frac{log(2/(p_{j(Q)}\delta))}{2n}} - \frac{\log\log c}{n} \text{ for some } Q \Bigg\} \leq \delta.$$

Here,

$$B(\kappa, S, Q) = \frac{1}{n} \sum_{i=1}^{n} \phi\left(y_i f(\boldsymbol{x}_i|Q)/\kappa\right) + \frac{4\,c}{\kappa} \sqrt{\frac{2\,A_{j(Q)}}{n}} + \frac{\log\log(2/\kappa)}{n}$$

and $j(Q) = \min\{j \mid \mathrm{D}[Q \,\|\, P] \leq A_j\}$.

The proof of the theorem is given in Appendix B.5. Note that the search for a minimiser κ is equivalent to an optimisation w.r.t. margin loss functions $\phi(\cdot/\kappa)$. Both $c > 1$ and the hierarchy (A_j, p_j) have to be chosen *a priori*. As long as they remain within sensible limits, the effect of their precise choice is insignificant: the bound value is typically dominated by the first two terms in $B(\kappa, S, Q)$. We may chose $c = 1.01,\ A_j = j\,n\,(\Delta A),\ p_j = 1/(j(j+1))$, where $\Delta A > 0$ is a grid size.

3.4.1.1 Comparing the PAC-Bayesian Theorems

Theorem 3.4 is quite different in form from Theorem 3.1 and the corresponding statement for the Bayes classifier mentioned in Section 3.2.5. A direct analytical comparison is difficult, because we would expect both theorems to show their merits only in "lucky" cases (which frequently occur in practice). In this section, we give some comparative arguments, pointing out a serious lack in tightness in the result of Meir and Zhang. An empirical comparison is presented in Section 3.5.6.

We are interested in the GP binary classification situation. Let $r_i = (\mu(\boldsymbol{x}_i) + b)/\sigma(\boldsymbol{x}_i)$. From (3.20) we know that

$$\hat{e}_{\text{Gibbs}} = \frac{1}{n} \sum_{i=1}^{n} \Phi(-y_i r_i).$$

It is also easy to see that $f(\boldsymbol{x}_i|Q) = 2\Phi(r_i) - 1$, so that

$$\frac{1}{n} \sum_{i=1}^{n} \phi\left(\kappa^{-1} y_i f(\boldsymbol{x}_i|Q)\right) = \frac{1}{n} \sum_{i=1}^{n} \phi\left(\kappa^{-1}(2\Phi(y_i r_i) - 1)\right).$$

The Bayes classifier makes a mistake whenever $y_i r_i \leq 0$, in which case we have $\phi(\kappa^{-1}(2\Phi(y_i r_i) - 1)) = 1$ independent of κ, while the contribution to the Gibbs empirical error can still be close to $1/2$ if $|r_i|$ is small. The reason why the Bayes variant often outperforms the Gibbs variant can only lie with patterns for which $|r_i|$ is fairly small, because Gibbs will with high probability make the same prediction as Bayes if $|r_i| \gg 0$. In these situations, among the patterns with small $|r_i|$ there are significantly more "on the right side", i.e. $y_i r_i > 0$. The contribution to the margin loss is smaller than the one to the Gibbs error once $\Phi(y_i r_i) > 1/(2-\kappa)$, and in this region the margin loss drops to zero quickly (slope κ^{-1}) while the Gibbs contribution is ≈ 0 only for $|r_i| > 2$ (see Figure 3.2).

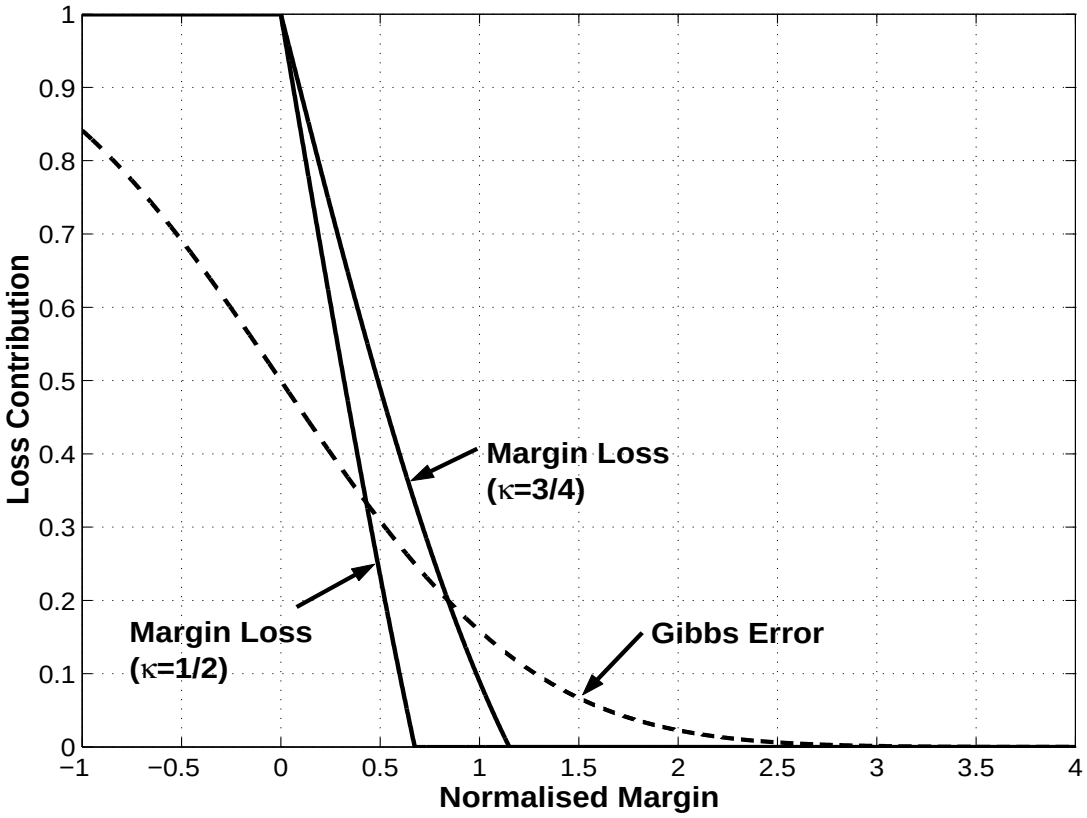

Figure 3.2: Contribution of single pattern to expected Gibbs error and Bayes margin loss respectively. The horizontal unit is the "normalised margin" $y_i r_i = y_i(\mu(\boldsymbol{x}_i) + b)/\sigma(\boldsymbol{x}_i)$.

The advantage of the margin loss over the expected Gibbs error grows with

smaller κ, but the other dominant term in the bound of Theorem 3.4 scales as $1/\kappa$. This second term poses a serious problem in the Bayes bound, since it scales as $(\mathrm{D}[Q \,\|\, P]/n)^{1/2}$ instead of $\mathrm{D}[Q \,\|\, P]/n$ in the Gibbs bound. This becomes an issue w.r.t. tightness if the empirical Bayes error and the empirical margin loss is much smaller than $1/2$, as will usually be the case in practice. It is interesting that it is exactly this case that the authors put forward in [121] to contrast their result with McAllester's. They point out rightly that one should rather focus PAC studies on the Bayes than on the Gibbs classifier, given that the former usually has much lower error rates in practice. However, their bound fails to exploit properly this case of low error rates far from $1/2$. In classical VC bounds which deal with zero-one loss, this problem is elegantly circumvented by bounding relative deviations such as $(p - \hat{p})/\sqrt{p}$ (see [196, 5]). A bound $(p - \hat{p})/\sqrt{p} \leq \varepsilon$ implies

$$p \leq \hat{p} + \frac{1}{2}\varepsilon^2 \left(1 + \sqrt{1 + 4\hat{p}/\varepsilon^2}\right),$$

so that the bound on the gap $p - \hat{p}$ is $O(\varepsilon)$ in the worst case $\hat{p} = 1/2$, but can be much smaller if $\hat{p}$ is small. It would be interesting to find out whether this technique carries over to the PAC-Bayesian theorem.[14] In fact, recently exponential concentration inequalities have been provided which exploit a small variance of the unknown process [21, 20], these could possibly serve as a substitute of the bounded difference inequality used in the present proof. Furthermore, the critical term drops below 1 only if $32\,\mathrm{D}[Q \,\|\, P] < n$, which may not happen at all in practice (see Section 3.5.6).

3.4.1.2 Towards a PAC-Bayesian Theorem for Regression

How could we obtain a PAC-Bayesian theorem for a regression model? We are not aware of the existence of a theorem comparable in tightness to Theorem 3.1 or Theorem 3.4 for the regression setting. Note that while Gibbs versions of Bayes-like classifiers often show acceptable performance in practice, a Gibbs variant of

[14] The proof of the zero-one loss situation does not carry over straightforwardly. Symmetrisation by a ghost sample and randomisation can be done, but it is not clear (to us) how to get rid of the margin loss function (in the proof of Theorem 3.4 this is done using a contraction principle, see Section B.5).

a regression estimator would not be sensible, since the estimate is required to be smooth. In this section, we present some thoughts towards a PAC-Bayesian theorem for regression, however without coming to a definite conclusion. Readers not interested in speculative material should jump to the next section. The details are given in Appendix B.5.1.

Recall the GP regression model from Section 2.1.2. The latent process $u(\boldsymbol{x}|\boldsymbol{w})$, *a priori* Gaussian with kernel K, is obscured by independent noise $P(y|u)$ before observation. We will use a non-negative loss function $\phi(\cdot)$ which is Lipschitz with maximum slope κ, and $\phi(0) = 0$. Loss is quantified as $\phi(y - f(\boldsymbol{x}))$, for example the empirical risk is

$$\frac{1}{n}\sum_{i=1}^{n}\phi\left(y_i - \mathrm{E}_{\boldsymbol{w}\sim Q}[u(\boldsymbol{x}_i|\boldsymbol{w})]\right). \tag{3.23}$$

Note that if ϕ is also convex (Huber or Laplace loss are examples of ϕ which are Lipschitz and convex, see Section A.6.1), we could obtain a bound on the risk from a bound on the risk of the not very useful, but theoretically possible Gibbs regression estimate:

$$\mathrm{E}_{(\boldsymbol{x}_*,y_*)}\left[\phi\left(y_* - \mathrm{E}_{\boldsymbol{w}\sim Q}[u(\boldsymbol{x}_*|\boldsymbol{w})]\right)\right] \leq \mathrm{E}_{(\boldsymbol{x}_*,y_*),Q}\left[\phi\left(y_* - u(\boldsymbol{x}_*|\boldsymbol{w})\right)\right].$$

However, such a bound would most probably be in terms of the empirical risk of the Gibbs estimate, which can be significantly larger than the empirical risk of the Bayes estimate.

We can try to move along the lines of Theorem 3.4. The first steps in the proof of Theorem 1 in [91] make use of Hoeffding's bounded differences inequality (see Section 2.2.2) which is not directly applicable to the regression setting with unbounded loss. Therefore, in order to prove a theorem in the regression setting, a different concentration argument would have to be used. To this end, it is probably necessary to make some sort of assumptions (such as bounded variance) on the distribution of the targets.

However, the remaining part of the proof, namely the bounding of the term $\mathrm{E}\|P_n - P\|_{\mathcal{G}_\phi}$ can be done, as is shown in Appendix B.5.1. Here, $\mathcal{G}_\phi$ consists of the functions $(\boldsymbol{x}, y) \mapsto \phi(y - \mathrm{E}_{\boldsymbol{w}\sim Q}u(\boldsymbol{x}|\boldsymbol{w}))$, $Q \in \mathcal{Q}_A$:

$$\mathrm{E}\|P_n - P\|_{\mathcal{G}_\phi} \leq \frac{4}{\kappa\sqrt{n}}\left(\mathrm{E}_S\left[\sqrt{2\,A\,(\operatorname{tr}\boldsymbol{K}/n)}\right] + \sqrt{\mathrm{E}\left[y^2\right]}\right). \tag{3.24}$$

Here, $\mathrm{E}[y^2]$ is an expectation over the data distribution. Furthermore, $n^{-1}\operatorname{tr}\boldsymbol{K}$ should concentrate under mild conditions on the data distribution and/or the kernel function. For example, if K is a stationary kernel (see Section 2.1.1), then $K(\boldsymbol{x},\boldsymbol{x}) = C$ for every $\boldsymbol{x}$, and $\operatorname{tr}\boldsymbol{K}/n = C$, the prior process variance. If we could show concentration of $\|P_n - P\|_{\mathcal{G}_\phi}$ under a boundedness condition on $\mathrm{E}[y^2]$, we would end up with a PAC-Bayesian theorem for the GP regression model.

3.5 Experiments

In this section, we present experiments testing instantiations of the PAC-Bayesian Theorem 3.1 for the Laplace GP Gibbs classifier in Section 3.5.2 and the sparse greedy GP (IVM) Gibbs classifier in Section 3.5.3 (these special cases of the bound are described in Sections 3.3.2 and 3.3.3). The results indicate that the bounds are very tight even for training samples of moderate sizes. In Section 3.5.4, we compare our bound to a state-of-the-art PAC compression bound for the IVM Bayes classifier, and to the same compression bound for the soft-margin support vector classifier in Section 3.5.4.1. In Section 3.5.5 we try to evaluate the model selection qualities of the PAC-Bayesian bound, again applied to the IVM Gibbs classifier. Finally, in Section 3.5.6 we strengthen some of the results by repeating experiments on a different setup and present comparisons with the bound of Meir and Zhang (see Section 3.4.1).

3.5.1 The Setup MNIST2/3

Let us describe the experimental design MNIST2/3 for most of the sections to come. This design applies in all sections but the last one (3.5.6). A real-world binary classification task was created on the basis of the well-known MNIST handwritten digits database[15] as follows. MNIST comes with a training set of 60000 and a test set of 10000 handwritten digits, represented as 28-by-28-pixel bitmaps, the pixel intensities are quantised to 8 bit values. First, the input dimensionality was reduced by cutting away a 2-pixel margin, then averaging intensities over

[15] Available online at *http://www.research.att.com/∼yann/exdb/mnist/index.html*.

3-by-3-pixel blocks, resulting in 8-by-8-pixel bitmaps. We concentrated on the binary subtask of discriminating twos against threes, for which a training pool of 12089 cases and a test set of $l = 1000$ cases were created.

The Gaussian process prior in our model was parameterised by a Gaussian (RBF) kernel with variance parameter $C > 0$ and an inverse squared length scale $w > 0$ (2.29). Since all the tasks we considered are balanced, we did not employ a bias parameter in the noise model.

The experimental setup was as follows. An experiment consisted of ten independent iterations. During each iteration, three datasets were sampled independently and without replacement from the training pool: a model selection (MS) training set of size n_{MS}, a MS validation set of size l_{MS} and a task training sample S of size n. Note that the latter set is sampled independently from the model selection sets, ensuring that the prior P in Theorem 3.1 is independent of the task training sample. Then, model selection was performed over a list of candidates for (w, C), where a classifier was trained on the MS training set and evaluated on the MS validation set (the MS score was the expected empirical error of the Gibbs classifier on the MS validation set). The winner was then trained on the task training set S and evaluated on the test set. Alongside, the upper bound value given by Theorem 3.1 was evaluated. The confidence parameter δ was fixed to 0.01.[16] We also quote total running times for some of the experiments. Where timing figures are given, these have been obtained on the same machine using the same implementation.

3.5.2 Experiments with Laplace GPC

The Laplace approximation framework for GP binary classification is described in Section 2.1.3, and the application of the PAC-Bayesian theorem to this method in Section 3.3.2. We used a logit noise model without a bias term, $P(y|u) = \sigma(y\,u)$, σ the logistic function. Note that both the computation of the relative entropy term and the expected empirical error require $O(n^3)$, and predictions

[16] Note that much smaller values of δ could have been used without altering the upper bound values significantly.

are $O(n^2)$ per pattern (we did not employ the sampling techniques from Appendix B.4). Note also that the complete kernel matrix $\boldsymbol{K}$ has to be evaluated and stored. Our implementation requires only one buffer of size n^2.

The specifications and results for the experiments of this section are listed in Table 3.1. For all these experiments, we chose model selection validation set size $l_{\mathrm{MS}} = 1000$ (recall that the test set is fixed with size $l = 1000$). Experiments #1 to #5 have growing sample sizes $n = 500, 1000, 2000, 5000, 9000$, the corresponding MS training set sizes are $n_{\mathrm{MS}} = 1000$ for experiments #2 to #5, and $n_{\mathrm{MS}} = 500$ for experiment #1. Note that $n_{\mathrm{MS}} < n$ in experiments #3 to #5 is chosen for computational feasibility, due to the considerable size of the candidate list for (C, w).

#	n	n_{MS}	emp	gen	upper	gen-bayes	time
1	500	500	0.036	**0.0469**	**0.182**	0.0339	14
			($\pm$0.0039)	($\pm$0.0015)	($\pm$0.0057)	($\pm$0.0023)	
2	1000	1000	0.0273	**0.036**	**0.131**	0.0274	67
			($\pm$0.0023)	($\pm$0.001)	($\pm$0.0041)	($\pm$0.0022)	
3	2000	1000	0.0243	**0.028**	**0.1091**	0.0236	91
			($\pm$0.0026)	($\pm$0.0013)	($\pm$0.0079)	($\pm$0.0029)	
4	5000	1000	0.0187	**0.0195**	**0.076**	0.0171	762
			($\pm$0.0016)	($\pm$0.0011)	($\pm$0.002)	($\pm$0.0016)	
5	9000	1000	0.0178	**0.0172**	**0.0706**	0.0158	3618
			($\pm$0.0012)	($\pm$0.0013)	($\pm$0.0037)	($\pm$0.0017)	

Table 3.1: Experimental results for Laplace GPC. n: task training set size; n_{MS}: model selection training set size. emp: expected empirical error; gen: expected generalisation error (estimated as average over test set). upper: upper bound on expected generalisation error after Theorem 3.1. gen-bayes: test error of corr. Bayes classifier. time: total running time per run (secs). Figures are mean and width of 95% t-test confidence interval.

Note that the resource requirements for our experiments are well within to-

day's desktop machines computational capabilities. For example, experiment #4 was completed in total time of about 12 to 13 hours, the memory requirements are around 250M. Now, for this setting both the expected empirical error and the estimate (on the test set) of the expected generalisation error lie around 2%, while the PAC bound on the expected generalisation error given by Theorem 3.1 is 7.6% — an impressive, highly non-trivial result on samples of size $n = 5000$. Our largest experiment #5 was done mainly for comparison with experiment #2 for IVM (see Section 3.5.3). The total computation time was 6 hours for each iteration, and the memory requirements are around 690M. We note a slight improvement in test errors as well as in the upper bound values (which now lie around 7%).

The "gen-bayes" column in Table 3.1 contains the test error that a *Bayes* classifier with the same approximate posterior as the Gibbs classifier attains (see Section 3.1.2). Note that it is not necessarily the best we could obtain for a Bayes classifier, because the model selection is done specifically for the Gibbs variant. In the Laplace GPC case we note that Bayes and Gibbs variants perform comparably well, although the Bayes classifier attains slightly better results and, as mentioned in Section 3.3.2, can be evaluated more efficiently. We include these results for comparison only: although the Gibbs result implies a bound on the generalisation error of the Bayes classifier (see Section 3.2.5), the link is too weak to render a sufficiently tight result.

3.5.3 Experiments with Sparse Greedy GPC

The sparse IVM approximation to GP binary classification is discussed in Section 4.4.1 and the corresponding PAC-Bayesian theorem in Section 3.3.3. In comparison with Laplace GPC, both training and evaluation are much faster now: $O(n\,d^2)$ and $O(d^2)$ respectively. The bound value can be computed in $O(n\,d^2)$. In our experiments here, the final active set size d was fixed *a priori*. Training was done in the same way as discussed in Section 4.8.1. For all experiments reported here, we chose MS training size $n_{\mathrm{MS}} = 1000$, MS validation size $l_{\mathrm{MS}} = 1000$ and $d_{\mathrm{MS}} = 150$. Note that in experiments which have the same $(n, n_{\mathrm{MS}}, l_{\mathrm{MS}})$ constel-

lation as Laplace GPC experiments, we use the same data subsets, in order to facilitate direct comparisons. The results are listed in Table 3.2.

#	n	d	emp	gen	upper	gen-bayes	time
1	5000	500	0.0154	**0.0207**	**0.067**	0.0084	16
			($\pm$0.0021)	($\pm$0.0015)	($\pm$0.0026)	($\pm$0.0014)	
2	9000	900	0.0101	**0.0116**	**0.0502**	0.0042	82
			($\pm$ 6.88e-4)	($\pm$ 5.49e-4)	($\pm$ 6.13e-4)	($\pm$ 8.79e-4)	

Table 3.2: Experimental results for sparse GPC. n: task training set size; d: final active set size. emp: expected empirical error; gen: expected generalisation error (estimated as average over test set). upper: upper bound on expected generalisation error after Theorem 3.1. gen-bayes: test error of corr. Bayes classifier. time: total running time per run (secs). Figures are mean and width of 95% t-test confidence interval.

Let us compare these results to the ones obtained for Laplace GPC. The sparse GPC Gibbs classifier trained with 5000 examples attains an expected test error of 2.1%, and the upper bound evaluates to 6.7%. While the former is the same as for the Laplace GPC variant, the latter is significantly lower. The ratio between upper bound and expected test error is 3.19, the ratio between gap bound and expected test error is 2.46. Note that experiment #1 for the sparse GPC was completed in total time of about 16 minutes — almost fifty times faster than the Laplace GPC experiment #4. It is interesting to observe that for this sample size, the results here are significantly better than for the full Laplace GPC on the same task[17] (experiment #5 in Section 3.5.2). Finally note that we did not try to optimise the final active set size d, but simply fixed $d = n/10$ *a priori*. An automatic choice of d could be based on heuristics which evaluate the error on

[17]Note that we are comparing two quite different ways of approximating the true posterior by a Gaussian: a Laplace approximation around the *mode* (which is different from the posterior *mean* — the "holy grail" of Bayesian logistic regression, see Section 2.1.3) and an approximation based on repeated moment matching (see Section 4.3). A more meaningful direct comparison would involve the cavity TAP method of [139] (see Section 4.3) which is, however, more costly to compute than the Laplace approximation.

the datapoints outside the active set (see Section 4.4.1).

The "gen-bayes" column in Table 3.2 serves the same purpose as the "gen-bayes" column in Table 3.1. In case of sparse greedy GPC, the results show that the Bayes classifier performs significantly better than the Gibbs variant, although the latter still attains very competitive results. A possible explanation for this difference, given that it cannot be observed for Laplace GPC, is obtained by inspecting the (C, w) kernel parameters values that are preferred by sparse greedy GPC. The parameter C is much larger for sparse GPC, i.e. the latent process has a larger *a priori* variance. This typically leads to an increase in the predictive variances, which in turn might introduce more sampling errors in the Gibbs predictions.

3.5.4 Comparison with PAC Compression Bound

In this section, we present experiments in order to compare our result for sparse GPC (Section 3.5.3) with a state-of-the-art PAC compression bound. Note that here, we employ *Bayes* GP classifiers instead of Gibbs GP classifiers: it would not be fair to compare our Gibbs-specific bound to an "artificially Gibbs-ified" version of a result which is typically used with Bayes-like "averaging" classifiers. A compression bound applies to learning algorithms which have some means of selecting a subsample S_I of size d from a sample of size n, such that the hypothesis they output is independent of $S_{\setminus I} = S \setminus S_I$, given S_I. More details are given in Appendix B.6, where we also state and prove the compression bound we are using here (Theorem B.2). Examples of compression schemes are the perceptron learning algorithm of [154], the support vector machine (SVM) and the sparse greedy IVM. The PAC compression bound of Theorem B.2 depends only on the training error for the remaining $n - d$ patterns $S_{\setminus I}$ outside the active set (called $\mathrm{emp}^{\setminus d}(S)$) and on d. We repeated the experimental setup used in Section 3.5.3 and employed the same dataset splits. The results can be found in Table 3.3.

For both experiments, $\mathrm{emp}^{\setminus d}(S) = 0$ was achieved in all runs, the compression bound is tightest in this case. Nevertheless, in experiment #1, the upper bound on the generalisation error is 30.5%, a factor of 50 above our estimate on the

#	n	d	emp	gen	upper
1	5000	500	0.0025	0.0058	0.3048
			(± 6.79e-4)	(±0.0015)	(± 0)
2	9000	900	0.0024	0.003	0.3041
			(± 4.25e-4)	(± 7.54e-4)	(± 0)

Table 3.3: Experimental results for PAC compression bound with sparse GP Bayes classifier. n: task training set size; d: final active set size. emp: empirical error (on full training set); gen: error on test set. upper: upper bound on generalisation error given by PAC compression bound. Figures are mean and width of 95% t-test confidence interval.

test set. The ratio is even worse for experiment #2. At least on this task, the PAC-Bayesian theorem produces much tighter upper bound values than the PAC compression bound. Note also that the compression bound is more restrictive, in that it applies to compression schemes only. On the other hand, the PAC-Bayesian theorem depends much more strongly on the model, prior assumptions and inference approximation algorithm, while the compression bound cannot differentiate between algorithms which attain the same level of compression and empirical error $\mathrm{emp}^{\setminus d}(S)$.

The reader may wonder why the generalisation errors here are slightly lower than the ones reported in Table 3.2. This should be due to the fact that in Section 3.5.3, we evaluated the Bayes classifier based on the hyperparameter values which have been optimised for the *Gibbs* variant, while here we performed model selection for the Bayes classifier explicitly.

3.5.4.1 Comparison with Compression Bound for Support Vector Classifiers

We can also compare our results for sparse GP Gibbs classifiers with state-of-the-art bounds for the popular soft-margin support vector machine (SVM). The latter is introduced in Section 2.1.6, where we also discuss similarities and differences to proper GP classification models. In the context here, it is important to note

that the SVM algorithm often produces rather sparse predictors. However, the degree of sparseness is not a directly controllable parameter, furthermore it is not an explicit algorithmic goal of the SVM algorithm to end up with a maximally sparse expansion. The aim is rather to maximise the "soft" minimal empirical margin (see Section 2.1.6). SVM is a compression scheme (see Section B.6), where d is the number of support vectors (vectors which are misclassified or lie within the margin tube around the decision hyperplane in feature space; see Section 2.1.6). Note that for SVM, we always have $\text{emp}^{\setminus d}(S) = 0$, since misclassified points are support vectors. Experimental results for SV classifiers and the PAC compression Theorem B.2 can be found in Table 3.4. Again, we employed the same dataset splits as used in Section 3.5.3.

#	n	emp	gen	upper	num-sv
1	5000	0.0016	0.0048	0.2511	370.8
		(± 9.49e-4)	(±0.0012)		(±10.71)
2	9000	0.0021	0.0036	0.213	529
		(± 7.67e-4)	(±0.0012)		(±29.12)

Table 3.4: Experimental results for PAC compression bound with SV classifiers. n: task training set size. emp: empirical error (on full training set); gen: error on test set. upper: upper bound on generalisation error given by PAC compression bound. Figures are mean and width of 95% t-test confidence interval.

In both experiments, a higher degree of sparsity is attained than the one chosen in the experiments above for sparse GPC (as mentioned above, we did not try to optimise this degree in the sparse GPC case), leading to somewhat better values for the PAC compression bound. However, the values of 25% (experiment #1) and 21% (experiment #2) are still by factors > 50 above the estimates computed on the test set, which is not useful in practice.

The compression bound applies to SVM, but is certainly not specifically tailored for this algorithm, since it does not even depend on the empirical margin distribution. Can we get better results with other SVM-specific bounds?

The margin bound of [177], commonly used to justify data-dependent structural risk minimisation for SVM, becomes non-trivial (i.e. smaller than 1) only for $n > 34816$ (see [73], Remark 4.33). The algorithmic stability bound of [22] does not work well for support vector classification either. In fact, the gap bound value converges to zero at some rate $r(n)$ (for $n \to \infty$) only if the variance parameter[18] C goes to zero at the same rate $r(n)$. If $r(n) = O(1/n)$ or $r(n) = O(\log n/n)$, this would correspond to severe over-smoothing. Herbrich and Graepel [73] use some older PAC-Bayesian theorems from [116] to prove a bound which depends on the minimal normalised empirical margin. This theorem applies to hard-margin SVC only and becomes non-trivial once the minimal normalised (hard) margin[19] is > 0.91, given that the feature space has dimension $> n$. Hard-margin SVMs tend to overfit on noisy real-world data with very small normalised margins at least on some points, and in practice the soft-margin variant is typically preferred. In a separate experiment using the same setup and dataset splits as in #1 of this section (i.e. training sample size $n = 5000$), but training hard margin SVMs without bias parameter, we obtained generalisation error estimates on the test set of gen $= 0.0056$ ($\pm$ 3.904e-5), minimum normalised margins of minmarg $= 0.0242$ ($\pm$ 2.813e-5) and generalisation upper bound values of upper $= 16.28$ ($\pm$ 4.7e-3) using the theorem of [73]. These results back the simple observation that the minimum normalised (hard) margin is not suitable as a PAC gap bound statistic and should probably be replaced by one which is more robust against noise, such as soft margin, sparsity degree or combinations thereof. All in all, and much to our surprise, we were not able to find any proposed "SVC-specific" bound which would be tighter on this task than the simple PAC compression bound of Theorem B.2 used above.

[18] In the SVM literature, it is common practice to separate C from the covariance kernel and write it in front of the sum over the slack variables. The parameter λ in [22] is $\lambda = 2/(Cn)$, and their gap bound behaves as $1/(n\lambda)$ as $n \to \infty$.

[19] If we view SVC as a linear method in a feature space induced by the covariance kernel, the minimal normalised margin is the arc cosine of the maximal angle between the normal vector of the separating plane and any of the input points mapped into feature space. A minimal normalised margin close to 1 means that *all* mapped input points lie within a double cone of narrow angle around the line given by the normal vector. For noisy data, such a situation is arguably quite unlikely to happen.

3.5.5 Using the Bounds for Model Selection

Can our results be used for model selection? In our opinion, this issue has to be approached with care. It seems rather obvious that a generalisation error bound should not be used for model selection on a real-world task if it is very far above reasonable estimates of the generalisation error on this task. This argument is discussed in more detail in Section 2.2.4. Even though the PAC-Bayesian theorem applied to GP Gibbs classifiers offers highly non-trivial generalisation error guarantees for the real-world task described in this section, they still lie by a factor > 3 above the estimates on the test set. In spite of this fact, we follow the usual conventions and present results of an experiment trying to assess the model selection qualities of our bound.

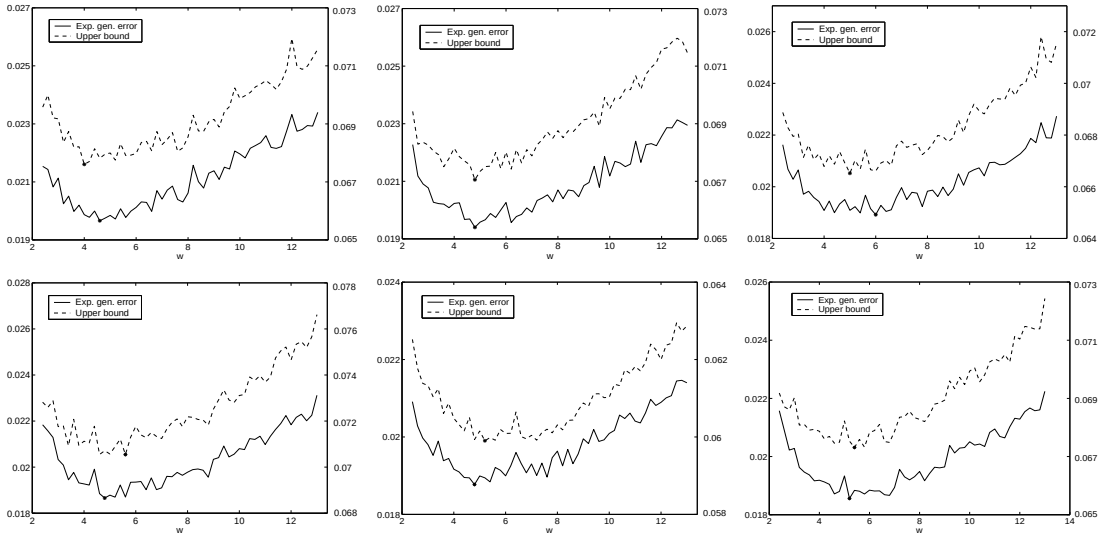

Figure 3.3: Comparing upper bound values with expected test errors. Solid line: expected test error (scale on left). Dashed line: upper bound value (*translated*, scale on the right). Respective minimum points marked by an asterisk.

Once more, we used sparse greedy GPC. The experiment consisted of six independent runs. We fixed $C = 120$ and used a grid of values for w. In each run, a sample S of size $n = 5000$ was drawn from the training pool, the method was trained for each configuration (C, w) (we fixed $d = 500$) and evaluated on the test set. The results are shown in Figure 3.3. We translated the upper bound

values towards the expected test errors by subtracting a constant (determined as 95% of the average distance between points on the two curves). In each graph, the scale on the left hand side is for the expected test error, the scale on the right hand side for the upper bound value. In Figure 3.4, we plot expected test errors (horizontal) vs. upper bound values (vertical). In this type of plot a mostly monotonically increasing relationship is what we would ideally expect. The dotted curves are lines $x + b$ with slope 1, where b is fitted to the corresponding solid curves by minimum least squares. The ordering of the six subplots is the same as in Figure 3.3.

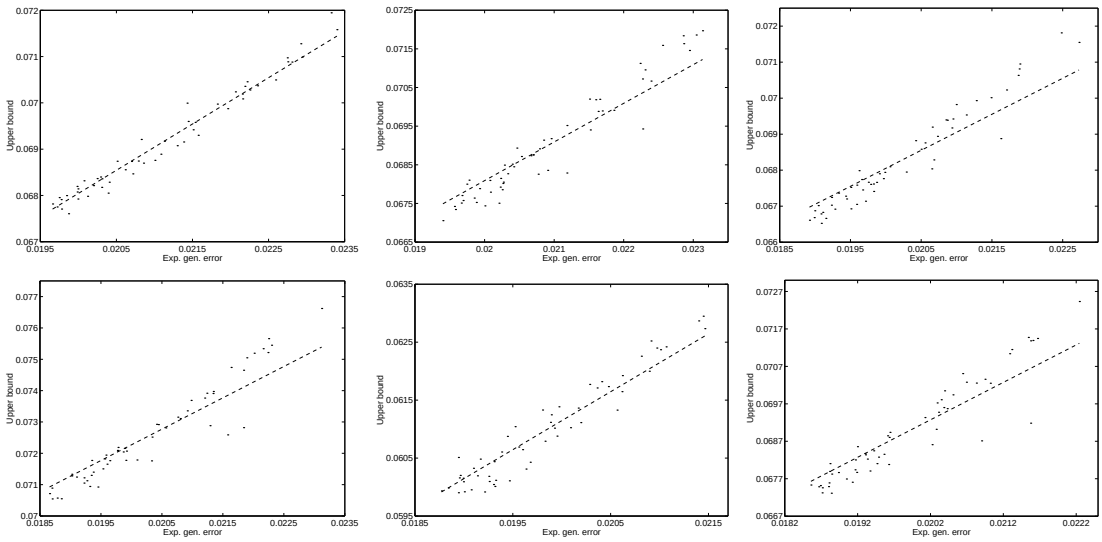

Figure 3.4: Comparing upper bound values (vertical axis) with expected test errors (horizontal axis). Dotted line: fitted regression line with slope 1.

In this particular experiment, there is a surprisingly good monotonically increasing linear correlation between upper bound values and expected test errors, and model selection based on minimising the upper bound value might have worked in this case. However, note that the constants we had to subtract from the upper bound curves in order to bring them close to the expected generalisation error estimates for visual inspection, were an order of magnitude larger than the range of variation of the individual curves shown in the plots. An important future direction of research would be to gain more understanding, both empir-

ically and analytically, of the phenomenon observed here: a slack term which is significantly larger than the expected test error, yet also much less variable than the latter over a range of hyperparameter values of interest (see also Section 3.2.6.2). Further experiments in Section 3.5.6 are however less conclusive and show that for other hyperparameters, the bound may not be predictive.

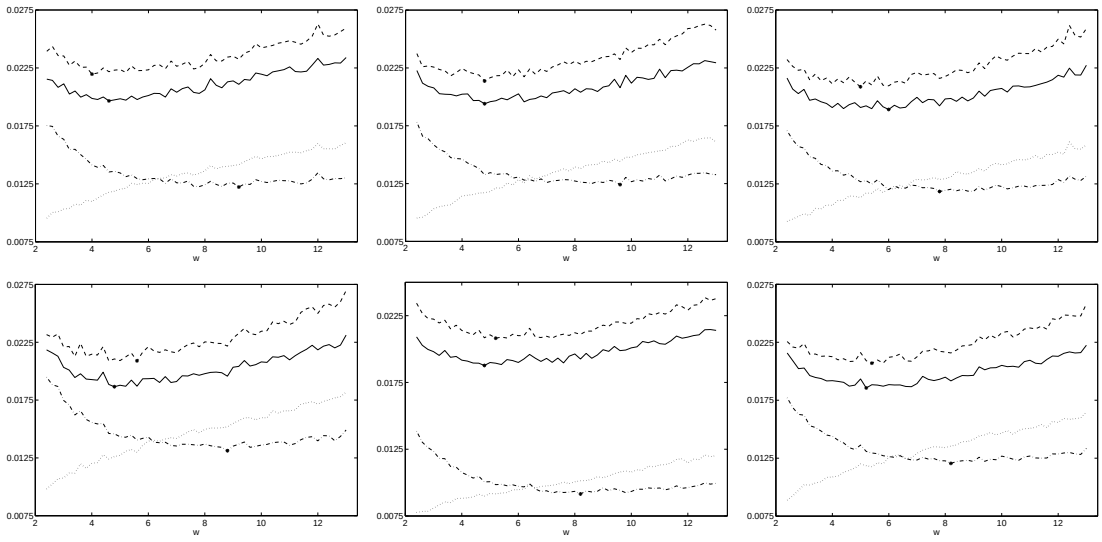

Figure 3.5: Comparing upper bound values with expected test errors (upper parts) and gap bound values with expected training errors (lower parts). Solid: expected test error (scale on left side). Dashed: upper bound value (*translated*). Dash-dotted: expected training error (scale on left side). Dotted: gap bound value (*translated*). Respective minimum points marked by an asterisk.

One might suspect that it is really only the expected empirical error which follows the expected test error closely, and that the difference between them remains fairly constant. After all, most of the points the empirical error is evaluated on are not in the active set. This argument of course ignores the fact that there is a dependence of the posterior on patterns outside the active set even for a compression scheme such as the IVM, namely they have *not* been chosen in favour of the active set. In theory, a PAC bound tells us that such dependencies have to be accounted for by an additional complexity term which is of rather crude union bound type in the PAC compression bound (Theorem B.2) and of a more refined

type in the PAC-Bayesian theorem. By splitting the upper bound curves above into expected empirical errors and gap bound values, we see that this extension is indeed necessary even if $d \ll n$. In Figure 3.5 we plotted all these curves together in common graphs, using the same ordering of runs as in the other graphs. In each subplot, the two curves at the top are the upper bound (dashed) and the expected test error (solid), while the two curves at the bottom are the expected empirical error (dash-dotted) and the gap bound (dotted). The scale is correct for both the expected empirical and test error curves, while the gap and upper bound curves are translated downwards by different constants. The scale for the latter two curves is omitted. Furthermore, the graphs in Figure 3.6 show that the linear correlation between expected training and test errors is poor, and indeed most of the graphs in Figure 3.5 exhibit over-fitting: model selection based on minimising the expected training error would choose too large values of w, corresponding to a too narrow kernel width.

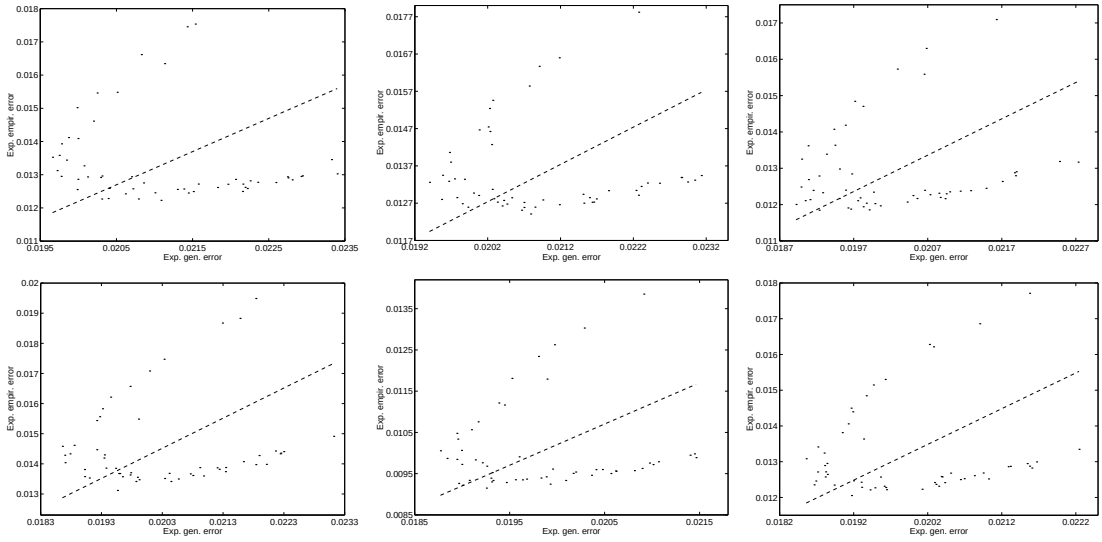

Figure 3.6: Comparing expected training errors (vertical axis) with expected test errors (horizontal axis). Dotted line: fitted regression line with slope 1.

3.5.6 Comparing Gibbs and Bayes Bounds

The aim of this section is twofold. First, in order to strengthen the results obtained in Sections 3.5.3 and 3.5.5 we repeat these experiments here using a different binary classification task. Second, we planned to compare the PAC-Bayesian theorem for Gibbs classifiers and its implications for the GP Bayes classifier against Theorem 3.4 obtained by Meir and Zhang. However, it turned out that the latter theorem does not render non-trivial bounds on this task, highlighting the problem pointed out in Section 3.4.1.1.

The setup MNIST8/9 we used here is similar to MNIST2/3 introduced in Section 3.5.1, but differs in the following points. We now consider the MNIST subtask of discriminating eights from nines. We used all available images from the official training set in our training pool (11800 cases) and tested on all images from the official test set ($l = 1983$), and the full 28-by-28-pixel bitmaps served as input points here.

We repeated the experiments of Section 3.5.3 for the single training sample size $n = 8000$, fixing the active size to $d = 800$, furthermore $n_{\mathrm{MS}} = l_{\mathrm{MS}} = 1000$ and $d_{\mathrm{MS}} = 150$. The outcome was emp = 0.0105($\pm$ 6.93e-4), gen = 0.0220($\pm$ 2.68e-4), upper = 0.0568($\pm$ 0.0012), quite in line with the earlier figures.

In order to compare the two PAC-Bayesian theorems for the GP Bayes classifier, we re-ran these experiments, but now using the Bayes instead of the Gibbs variant for model selection.[20] As noted in Section 3.2.5, we can use twice the value of the Gibbs bound as an upper bound for the Bayes classifier, however this is unsatisfactory given the apparent superior performance of the Bayes classifier on this task. When we tried to compare these results against the bound of Theorem 3.4 of Meir and Zhang, we ran into the problem mentioned in Section 3.4.1.1: the term $4c\kappa^{-1}(2\mathrm{D}[Q \,\|\, P]/n)^{1/2}$ was larger than one even for $\kappa = 1$, in all ten runs. In this situation, the bound cannot give non-trivial results for *any* prior configuration. We suspect that even in cases where the bound becomes non-trivial, the scaling with the *square root* of $\mathrm{D}[Q \,\|\, P]/n$ will render it very sub-

[20] The selection results were quite different. The Gibbs variant preferred larger C values, while the Bayes variant required larger w values.

optimal in practice. Note that this task, with $n = 8000$, is at the upper range of sample sizes we considered in our experiments. From these experimental findings, we conclude that the problem of a practically satisfying PAC-Bayesian bound for GP Bayes classifiers is not resolved yet.

Given the results in Section 3.5.5, the PAC-Bayesian bound seems quite suitable for model selection due to an excellent monotonically increasing linear correlation between expected test errors and upper bound values for different values of w. We repeated these experiments on MNIST8/9 with $n = 8000$, resulting in Figure 3.7.

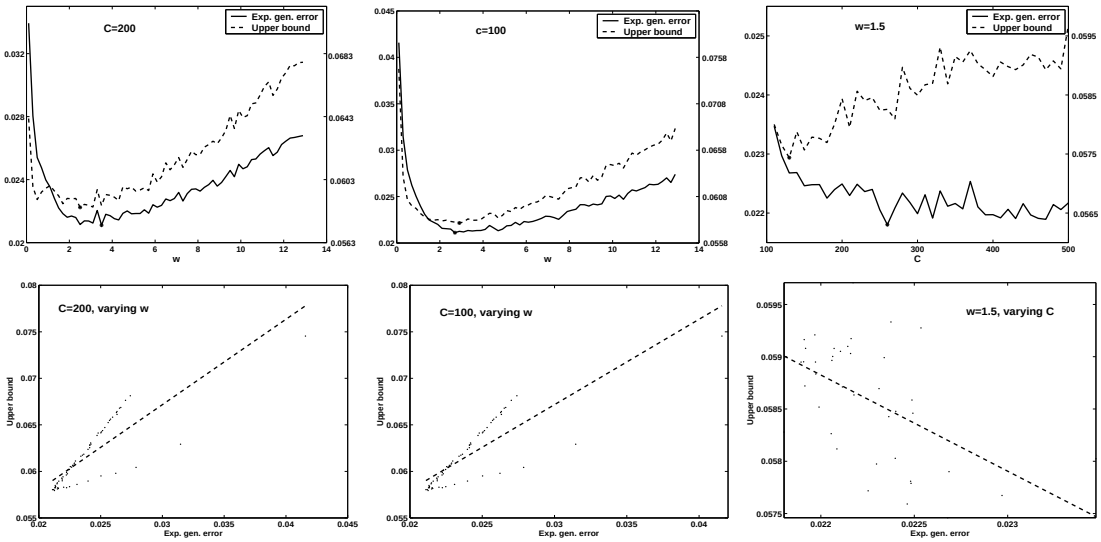

Figure 3.7: Comparing expected test errors with upper bound values on MNIST8/9, $n = 8000$. Upper row: hyperparameter values vs. test errors (scale left) and translated bound values (scale right). Lower row: test errors vs. bound values (dotted: fitted regression line). Left: varying w with $C = 200$. Middle: varying w with $C = 100$. Right: varying C with $w = 1.5$.

These results show that the upper bound value can have shortcomings as "predictor" for the expected test error. The plots in the left and middle columns are for fixed $C = 200$, $C = 100$ and varying w. Some values bail out of the otherwise linear relationship, notably the upper bound underestimates the sharp increase in test error for too small w. Recall that small w translate into large length scale and

therefore very smooth discriminant functions (decision boundaries). A preference for over-smooth solutions hints towards the bound placing too much weight on the complexity penalty. The fact that for large w values the bound grows faster than the expected test error also points in this direction. The right column plots show that the upper bound fails to predict the test error for varying C and fixed $w = 1.5$, again favouring over-smooth solutions (C is the prior process variance). These findings somewhat back the objections we raise in Section 2.2.4, showing that model selection based on current PAC bounds is risky at best.

3.6 Discussion

In this chapter, we have shown how to apply McAllester's PAC-Bayesian theorem to approximate Bayesian Gaussian process classification methods. The generic bound applies to a wide class of GP inference approximations, namely all that use a GP to approximate the predictive process (see Sections 2.1.3 and 4.6 for references). We have simplified the original proof of the PAC-Bayesian theorem considerably, pointing out convex duality (see Section A.3) as the principal underlying technique. We have also shown how to generalise the theorem to multi-class settings and the problem of bounding linear functions of the confusion matrix. Although in this work, we have focused on Gaussian process models, the PAC-Bayesian theorem can in principle be applied to any approximate Bayesian classification technique. The terms the bound depends upon typically fall out as simple byproducts of the inference approximation.

Additional conclusions and suggestions for future work can be found in Section 5.1.

Chapter 4

Sparse Gaussian Process Methods

When approaching a statistical learning problem with a non-parametric method, we truly let the data speak. Prior knowledge about the problem is used to smooth the data for visualisation and prediction, but there is no need to constrain the data generation process to be compatible with a pre-built model. In general, the principal drawback of non-parametric methods in practice is the heavy scaling with the size of the training sample, since what we have learned is represented by the virtually uncompressed sample itself. Predictions require associations of the test point with all training points, as does incremental learning. This is generally infeasible for very large samples. In sparse approximations, we try to filter out and discard redundant points from the sample, where the notion of redundancy depends on our current belief about the source. In this chapter, we clarify the nature of such approximations to Gaussian process inference and present some new schemes which employ greedy forward selection to filter the sample. We show that these schemes improve on the scaling of their non-sparse counterparts by orders of magnitude for large samples, thereby widening the range of problems which can now be attacked successfully using Gaussian process models.

The structure of this chapter is as follows. The introduction in Section 4.1 contrasts parametric and non-parametric methods and sets out our goals. In Section 4.2, we develop some general theory behind sparse GP approximations, namely the central notion of likelihood approximations and a discussion of greedy selection criteria borrowed from active learning to meet our sample subset selec-

tion requirements. The schemes we focus on here are sparse variants of the expectation propagation (EP) algorithm for GP models, which is introduced in Section 4.3. Two schemes for conditional sparse inference are developed in Section 4.4: the simple and fast IVM and the more accurate PLV. Approximate inference techniques are incomplete without dealing with the model selection issue, which we do in Section 4.5. The field of sparse GP approximations has received a lot of attention recently, relations to other work are clarified in Section 4.6. Finally, we present empirical studies in Section 4.8: binary classification and regression estimation with IVM and regression estimation with PLV. We close the chapter with a discussion in Section 4.9.

Parts of the results presented here have been obtained in collaborations with other researchers and have previously been published in conference proceedings, as is detailed in Section 1.1.

4.1 Introduction

Bayesian inference is certainly a very appealing principle, based on simple consistent rules at the core (see Section A.6). However, the computational burden is more often than not intractable, and the challenge is to find suitable approximations tailored to the particular situation at hand. As we have seen in Section 2.1.3, schemes for exact or approximate Bayesian inference in GP models typically suffer from heavy scaling with the number n of training points: $O(n^3)$, or at least $O(n^2)$ time for training, $O(n^2)$ memory for training, finally $O(n)$ or even $O(n^2)$ memory to represent the predictor and at least $O(n)$ time for each prediction on a test point. At least the latter figures are characteristic for *non-parametric methods* which use the training data in uncompressed form to represent the belief. In contrast, *parametric methods* such as logistic regression (see Section 2.1.2) or multi-layer perceptrons typically have $O(n)$ training time and memory as well as test time requirements independent of n. They represent the information in the training sample which actually affects the belief by an empirical statistic of size *independent* of n, which is why the belief can be parameterised in a way which

does not scale with n. A further advantage of parametric methods is that the belief can typically be updated fairly easily (in time independent of n) when new data comes in. A drawback of this approach is that it is often hard to come up with a sensible model based on some hard-wired statistics when confronted with a not very well-understood source, and inferences may be biased by artifacts and shortcomings of the parametric model which have little to do with the actual observations. Furthermore, while most non-parametric methods are universally consistent, i.e. will represent the true source in the limit of infinite data, this is typically not the case for parametric methods (if the size of the sufficient statistic does not grow with n). Finally, the relationship between parameters and represented distributions can be very complicated (e.g., for a multi-layer perceptron), in which case current approximation techniques to Bayesian inference can give very poor results or may be very difficult to handle and time-consuming.

Sparse approximations to GP models in a sense try to combine the flexibility and simplicity of non-parametric methods with the favourable scaling of parametric ones. Instead of *a priori* fixing the statistics which are applied to the training sample, they try to directly approximate the inference for a full GP model using a sparse representation for the posterior belief whose size can be controlled independently of n. By doing so, they nevertheless try to retain desirable properties of the scheme based on the full GP model. For example, the sparse approximations discussed here all retain the property that the predictive posterior process for the latent outputs is again Gaussian.

4.2 Likelihood Approximations and Greedy Selection Criteria

In this section, we show how sparse GP approximations can be understood as likelihood approximations. We also introduce a number of greedy selection criteria and discuss their potential significance for sparse GP techniques.

4.2.1 Likelihood Approximations

As we have motivated in Section 4.1, a sparse approximation to GP inference should lead to training time and memory requirements which scale essentially linearly in n, the number of training points, yet ideally retain the desirable property of full inference schemes, namely that the predictive posterior process $P(u(\cdot)|S)$ is again Gaussian (see Section 2.1.3). In this Section, we discuss a general framework for obtaining such approximations.

In order to retain the predictive posterior GP property, it seems necessary to follow the same procedure as in the general case (see Section 2.1.3): approximate $P(\boldsymbol{u}|S)$ by a Gaussian $Q(\boldsymbol{u}|S)$, then define the approximate predictive process via (2.10). Of course, it would be possible in principle to replace the conditional prior process $P(u(\cdot)|\boldsymbol{u})$ by a different GP depending on S and still obtain a Gaussian $Q(u(\cdot)|S)$. However, such an approximation would violate the property $u(\cdot) \perp S \,|\, \boldsymbol{u}$ of the true process, which does not seem sensible.

If the true posterior $P(\boldsymbol{u}|S) \propto P(S|\boldsymbol{u})P(\boldsymbol{u})$ is approximated by a Gaussian, we have $Q(\boldsymbol{u}|S) \propto Q(S|\boldsymbol{u})P(\boldsymbol{u})$, where $Q(S|\boldsymbol{u})$ as a function of $\boldsymbol{u}$ is an unnormalised[1] Gaussian (Definition A.10). Now, suppose that $Q(S|\boldsymbol{u})$ really only depends on a subset $\boldsymbol{u}_I$ of the components of $\boldsymbol{u}$, where $I \subset \{1, \ldots, n\}$, $|I| = d \ll n$. In this case, for the prediction at a new datapoint $\boldsymbol{x}_*$, we have

$$Q(u_*|\boldsymbol{x}_*, S) = \mathrm{E}_{Q(\boldsymbol{u}|S)}\left[P(u_*|\boldsymbol{u})\right] = \mathrm{E}_{Q(\boldsymbol{u}_I|S)}\left[P(u_*|\boldsymbol{u}_I)\right],$$

since

$$P(u_*|\boldsymbol{u})Q(\boldsymbol{u}|S) = P(u_*|\boldsymbol{u})P(\boldsymbol{u}_{\setminus I}|\boldsymbol{u}_I)Q(\boldsymbol{u}_I|S) = P(u_*, \boldsymbol{u}_{\setminus I}|\boldsymbol{u}_I)Q(\boldsymbol{u}_I|S).$$

Thus, if I and $Q(S|\boldsymbol{u}_I)$ are given, it is straightforward to compute the marginal $Q(\boldsymbol{u}_I|S)$, after which predictions can be done in $O(d^2)$, requiring perhaps a precomputation of $O(d^3)$ (see Section 2.1.3).

On the other hand, the Gaussian posterior approximation can always be written as

$$Q(\boldsymbol{u}|S) = N(\boldsymbol{u}|\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}), \quad \boldsymbol{A}^{-1} = \boldsymbol{K}^{-1} + \boldsymbol{\Pi},$$

[1] We allow for $\int Q(S|\boldsymbol{u})\, d\boldsymbol{u}$ not to exist, as long as the resulting posterior can be properly normalised.

where $\boldsymbol{\Pi}$ is symmetric. We can write $\boldsymbol{\Pi} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$ where $\boldsymbol{D} \in \mathbb{R}^{d,d}$ is diagonal and full rank. In general, neither $\boldsymbol{K}$ nor $\boldsymbol{A}$ has a special structure which could be used to perform inference in $O(n)$.[2] Furthermore, write $\boldsymbol{r} = \boldsymbol{U}\boldsymbol{v} + \boldsymbol{n},\ \boldsymbol{U}^T\boldsymbol{n} = 0$. Plugging this into (2.14), we end up with

$$\begin{aligned} \mu(\boldsymbol{x}_*) &= \boldsymbol{k}_*^T\boldsymbol{n} + \boldsymbol{k}_*^T\boldsymbol{U}\boldsymbol{\beta}, \\ \sigma^2(\boldsymbol{x}_*) &= K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \left(\boldsymbol{U}^T\boldsymbol{k}_*\right)^T \left(\boldsymbol{D} + \boldsymbol{U}^T\boldsymbol{K}\boldsymbol{U}\right)^{-1} \left(\boldsymbol{U}^T\boldsymbol{k}_*\right), \end{aligned} \tag{4.1}$$

where $\boldsymbol{k}_* = (K(\boldsymbol{x}_*, \boldsymbol{x}_i))_i$ and $\boldsymbol{\beta} \in \mathbb{R}^d$.

We see that in order to obtain an expression for the predictive variance which requires the evaluation of d components of $\boldsymbol{k}_*$ only, we need $\boldsymbol{U} = \boldsymbol{I}_{\cdot,I}\boldsymbol{E}$ for some index set I of size d and some $\boldsymbol{E} \in \mathbb{R}^{d,d}$. The exact conditions for a sparse $\mu(\boldsymbol{x}_*)$ are less clear, yet it seems that less restrictive constraints than $\boldsymbol{n} = \boldsymbol{0}$ and $\boldsymbol{r} = \boldsymbol{I}_{\cdot,I}\boldsymbol{v}$ are not feasible to work with. Now, note that $\boldsymbol{\Pi}$ is exactly the precision matrix in the Gaussian likelihood approximation $Q(S|\boldsymbol{u})$, and the constraints $\boldsymbol{U} = \boldsymbol{I}_{\cdot,I}\boldsymbol{E},\ \boldsymbol{r} = \boldsymbol{I}_{\cdot,I}\boldsymbol{v}$ are equivalent to $Q(S|\boldsymbol{u})$ depending on $\boldsymbol{u}_I$ only. Thus, given $Q(u(\cdot)|\boldsymbol{u}) = P(u(\cdot)|\boldsymbol{u})$, a restricted likelihood approximation is necessary for achieving the desired scaling in general.

It is interesting to note that if a non-sparse predictor (requiring all of $\boldsymbol{k}_*$) is acceptable but training has to scale as $O(n\,d^2)$, the restriction $\boldsymbol{r} = \boldsymbol{I}_{\cdot,I}\boldsymbol{v}$ is not strictly required, as is shown in Section C.2.2. However, it follows from (4.1) that the additional d.o.f.'s cannot influence the predictive variance approximations.

How could we choose a suitable likelihood approximation for a given model? As an example, note that the EP approximation to a general GP model with completely factorised likelihood (as discussed in Section 4.3) implies a likelihood approximation with a diagonal precision matrix $\boldsymbol{\Pi} = \mathrm{diag}(\pi_i)_i$ (so do a range of other GP approximations, see Section 2.1.3 where $\boldsymbol{\Pi}$ is called $\boldsymbol{D}$). A straightforward idea for a sparse likelihood approximation is simply to clamp most of the π_i to zero. We develop the corresponding sparse method in Section 4.4.1. In general, the use of non-diagonal $\boldsymbol{\Pi}$ may be preferable to correct for errors made by the sparse approximation. The sparse on-line method of [40] (see Section 4.6)

[2] In spline smoothing with very low-dimensional input spaces (see Section 2.1.5), the relevant matrices have a banded sparsity pattern which can be used to perform inference very efficiently.

is one of the most efficient examples. But what is an *optimal* form for a sparse likelihood approximation? Optimality is only defined relatively, yet the relative entropy (Definition A.4) between the corresponding posteriors is certainly a sensible criterion.

Lemma 4.1 ([40]) *Let $P(S|\boldsymbol{u})$ be some likelihood and $P(\boldsymbol{u})$ some prior, such that the posterior $P(\boldsymbol{u}|S)$ exists. None of them need to be Gaussian. Consider the set $\mathcal{F}$ of positive functions $Q(S|\boldsymbol{u}_I)$ of $\boldsymbol{u}_I$ with the property that $Q(\boldsymbol{u}|S) \propto Q(S|\boldsymbol{u}_I)P(\boldsymbol{u})$ exists. We have*

$$\operatorname*{argmin}_{Q(S|\boldsymbol{u}_I)\in\mathcal{F}} \mathrm{D}\left[P(\boldsymbol{u}|S) \,\|\, Q(\boldsymbol{u}|S)\right] \propto \mathrm{E}_P\left[P(S|\boldsymbol{u}) \,|\, \boldsymbol{u}_I\right] \tag{4.2}$$

and

$$\operatorname*{argmin}_{Q(S|\boldsymbol{u}_I)\in\mathcal{F}} \mathrm{D}\left[Q(\boldsymbol{u}|S) \,\|\, P(\boldsymbol{u}|S)\right] \propto \exp\left(\mathrm{E}_P\left[\log P(S|\boldsymbol{u}) \,|\, \boldsymbol{u}_I\right]\right). \tag{4.3}$$

If $P(\boldsymbol{u})$ is Gaussian and $P(S|\boldsymbol{u})$ is an unnormalised Gaussian w.r.t. $\boldsymbol{u}$, i.e. $P(S|\boldsymbol{u}) = N^U(\boldsymbol{u} \,|\, \boldsymbol{b}, \boldsymbol{\Pi})$, then $Q(S|\boldsymbol{u}_I) \propto N^U(\mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I] \,|\, \boldsymbol{b}, \boldsymbol{\Pi})$ is a maximiser for (4.3), where $\mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I]$ is the conditional mean under $P(\boldsymbol{u})$. In this case, the maximiser of (4.2) also is an unnormalised Gaussian, but is parameterised in terms of a d-by-d block of the inverse covariance matrix of $P(\boldsymbol{u})$.

The proof is given in Appendix C.2.1. While the optimal likelihood approximation in the sense of minimal $\mathrm{D}[P(\boldsymbol{u}|S) \,\|\, Q(\boldsymbol{u}|S)]$ is intractable in the context of sparse methods, the optimal approximation for the criterion $\mathrm{D}[Q(\boldsymbol{u}|S) \,\|\, P(\boldsymbol{u}|S)]$ *has* a tractable form: the conditional mean is $\mathrm{E}_P[\boldsymbol{u}|\boldsymbol{u}_I] = \boldsymbol{K}_{\cdot,I}\boldsymbol{K}_I^{-1}\boldsymbol{u}_I$, which can be computed by inverting $\boldsymbol{K}_I$ only. If we define $\boldsymbol{P} = \boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,\cdot}$, then $\mathrm{E}_P[\boldsymbol{u}|\boldsymbol{u}_I] = \boldsymbol{P}^T\boldsymbol{u}_I$, which can be seen as a reconstruction from an orthogonal projection and has been termed *KL-optimal projection* in [38]. In Section 4.4.2, we show how KL-optimal projections can be combined with expectation propagation to obtain a powerful sparse approximation method for GP inference.

4.2.2 Greedy Selection Criteria

A likelihood approximation most fundamentally depends on the active set $I \subset \{1, \ldots, n\}$, and a central part of any sparse approximation algorithm is the proce-

dure for selecting an appropriate I. Finding a globally optimal I (where "optimal" is defined in any non-trivial way) is a combinatorial problem, so heuristics will have to be used. We could pick I at random, but this will in general perform poorly on difficult classification tasks if $d \ll n$. Due to resource limitations, we will also have to restrict ourselves to methods which do not use active sets of size $> d_{max}$ (say) at any time, for example starting with $I = \{1, \ldots, n\}$ and shrinking to a desired size is not an option. Finally, we only consider methods which change the size of I by at most 1 in each iteration. Advantages include that updates of the posterior approximation can be done easily, and that scoring-based methods would have to evaluate many more candidates if updates were done in larger blocks. Amongst such methods, greedy forward selection is the simplest: elements are included once and for all, without the option to remove them later. Methods which allow for removals may result in better final selections, but are typically slower than forward selection and in our context are also more likely to cause numerical problems.

An optimal greedy selection criterion would lead to an active set I such that the decision boundary of the final predictor is close to the optimal one, or at least close to the one of a corresponding non-sparse predictor. The shape of the boundary depends most on the patterns close to it, as well as on mis-classified ones. Of course, we do not have the option to train a non-sparse predictor and then identify those points, but we can greedily approximate this goal by including points which are judged most uncertain (or even mis-classified) by the current sparse predictor, in other words prefer patterns which most contradict what we have learned so far. This idea has been widely used before and is related to *active learning* (or *sequential design*). In this scenario, we assume that a massive amount of i.i.d. unlabelled data (covariates) is available, from which we are required to select a training sample as small as possible to learn a classification or regression function. One can distinguish two different problems, depending on whether the labels are available at the time of selection or they are supplied only *after* each inclusion decision has been made. The former, the one we are facing, is a practical one: training on all data is prohibitively expensive. It is also somewhat simpler

than the latter query-based one which has been studied in [53, 108, 174, 58, 31]. An example for the former subset selection scenario is [143]. In [58] it is shown that a representative unlabelled sample is necessary for successful fast learning, because otherwise we would typically end up requesting labels at points which are extremely unlikely to occur (the "Achilles' heel" mentioned in [108] is related to this problem). Methods from these two regimes are often related, since a query-based scheme can be derived from a subset selection scheme by taking an additional expectation over the unknown label (for example by sampling based on the current predictor). Indeed, the criteria we consider here are the ones suggested in [108] with the expectation over the chosen target left away.

Let $Q(\boldsymbol{u})$ be our current belief, based on having included patterns in I, and let $Q_i^{new}(\boldsymbol{u})$ be the updated belief after including pattern i additionally. We can measure the amount of information gained by including i by the decrease in entropy

$$\Delta_i^{\mathrm{Ent}} = \mathrm{H}[Q_i^{new}] - \mathrm{H}[Q] \tag{4.4}$$

or by the negative relative entropy

$$\Delta_i^{\mathrm{Info}} = -\mathrm{D}[Q_i^{new} \,\|\, Q]. \tag{4.5}$$

The latter has the interpretation of reduction in coding loss when switching from a wrong model Q of a source Q_i^{new} to the correct one (see Section A.3). The entropy reduction quantifies by how much the posterior spread shrinks due to the new inclusion, while the relative entropy more generally quantifies changes in the posterior. The latter is especially large if the new Q_i^{new} puts significant weight into low-density regions of Q. Note that in other work on active learning [174, 58] a fast shrinkage of the "version space" is the main aim.[3] In a certain sense, the posterior entropy can be seen as proxy for version space volume in the Bayesian context. MacKay [108] notes that if Q and Q_i^{new} are Bayesian posteriors for S_I and $S_{I'}$ respectively, $I' = I \cup \{i\}$, then $\mathrm{E}[-\Delta_i^{\mathrm{Ent}}] = \mathrm{E}[-\Delta_i^{\mathrm{Info}}]$, where the expectation is over $P(y_i|\boldsymbol{x}_i, S_I)$. Nevertheless, they can be quite different as functions of y_i, especially in the crucial case of y_i being different than what we would expect

[3] The version space collects all hypotheses (from a given space) which are consistent with all data so far. Its volume is measured relative to the full hypothesis space (see [70]).

from $P(y_i|\boldsymbol{x}_i, S_I)$. We will refer to Δ_i^{Ent} as *differential entropy score* and to Δ_i^{Info} as *(instantaneous) information gain score*. Note that both scores are typically negative and lower values are more desirable.

While the criteria (4.4) and (4.5) have been widely used, it is interesting to note that while it can be very hard to obtain a useful approximation for them in the context of nonlinear parametric methods such as multi-layer perceptrons (MLPs), they are straightforward to compute in the case of Gaussian process models with the usual Gaussian approximations (see Sections 4.4.1 and 4.4.2.1). For example, in the case of MLPs Gaussian posterior approximations can be poor, and selection based on the information criteria which take these approximations as the truth can suffer badly. In the GP case, the Gaussian approximations are often quite good, and the criteria (4.4) and (4.5) can be evaluated without having to make further approximations.[4] Using the criteria for query-based active learning in GP models should be almost equally simple and is subject to future work. O'Hagan [135] is an early paper discussing optimal design for GP models, however the assumption that a large pool of unlabelled points is available is not made, rather the input distribution is considered known. Tong and Koller [193] give a method for active learning in SVMs (see Section 4.6).

We conclude this section by stressing an important point for subset selection methods: whatever criterion we use to score inclusion candidates i, we have to be able to evaluate it fast, indeed significantly faster than computing Q_i^{new} itself. As mentioned above, if we had the time to train on all available data or a significant fraction thereof, subset selection would not be required. In the presence of an *expensive* selection criterion, all we can do in each iteration is to draw a very small subsample of the pool and pick the top-scorer from there. Even so, the training time will then be strongly dominated by the score evaluations, in that the dominant cost scales with the size of the subsamples. Looking at (4.4) or (4.5), it seems that Q_i^{new} is required to compute them, but this need not be the case (see Section 4.4.1). If it is indeed, we recommend to look for a cheaper approximation to these criteria (a general approximation is suggested in Section 4.4.2.1). One

[4] Cohn et. al. [31] discuss active learning with mixtures of Gaussians and locally weighted linear regression, for which the criteria can be computed equally easily.

might conjecture that given two sensible criteria, a cheap and an expensive one, the latter should always be preferred if time is not an issue. This intuition is wrong in general, as the simple example in Appendix C.2.3 shows.

4.3 Expectation Propagation for Gaussian Process Models

In this section, we introduce the *expectation propagation (EP)* scheme [125, 124] for approximate inference in Gaussian process models. The parametric representations and update equations of EP are at the heart of the sparse GP algorithms to be discussed further below. The general EP scheme for exponential families is described in Appendix C.1.1, and we urge the reader to glance over the material there. In short, EP builds on the following idea. An intractable (posterior) belief which can be written as normalised product of positive sites, is to be approximated by a member of an exponential family $\mathcal{F}$. This is done by replacing each site which is not in $\mathcal{F}$ by a site approximation from the corresponding unnormalised family $\mathcal{F}^U$ (Definition A.7). The site parameters of these approximations are updated sequentially and iteratively. An EP update (Definition C.2) exchanges a site approximation against the true site to obtain a tilted exponential distribution (Definition A.6) outside of $\mathcal{F}$. This distribution is projected back into $\mathcal{F}$ preserving the moments w.r.t. which $\mathcal{F}$ is defined. Whether or not this is feasible[5] depends crucially on the structure of the sites. The so-called cavity distribution $Q^{\setminus i}$ is proportional to the current belief Q divided by the i-th site approximation, and the tilted distribution is proportional to $Q^{\setminus i}$ times the i-th site.

Suppose now we are given a Gaussian process model with completely factorised likelihood, in the sense that each observed output y_i depends on one corresponding latent output $u_i = u(\boldsymbol{x}_i)$ only. In this case, we have seen in Section 2.1.3 that inference (conditioned on hyperparameters) requires merely the

[5]Note that the direct projection of the posterior belief onto $\mathcal{F}$ is considered intractable, otherwise an iterative approximation such as EP is not required.

computation of the posterior $P(\boldsymbol{u}|S)$, where $\boldsymbol{u} = (u_1, \ldots, u_n)^T$ collects the latent outputs at the training points $S = \{(\boldsymbol{x}_i, y_i)|i = 1, \ldots, n\}$. In this case, we would like to approximate the posterior

$$P(\boldsymbol{u}|S) \propto P(\boldsymbol{u}) \prod_{i=1}^{n} P(y_i|u_i),$$

where[6] the prior is $P(\boldsymbol{u}) = N(\boldsymbol{0}, \boldsymbol{K})$, $\boldsymbol{K}$ the kernel matrix over the training set, which matches the situation of Section C.1.1 if $\mathcal{F}$ is the family of Gaussians over $\boldsymbol{u}$ and $t_i(u_i) = P(y_i|u_i)$. It is clear that if all sites are Gaussian (like in GP regression with Gaussian noise, see Section 2.1.2), the posterior can be worked out directly without the need for an EP approximation. Note that each site depends on one component of $\boldsymbol{u}$ only, which allows us to use the locality property of EP (Lemma C.1). It is in principle straightforward to develop EP for Gaussian process models in which each site depends on a small number of components of $\boldsymbol{u}$, which is the case for example in multi-class GP classification (see Section 2.1.2). For another example, see [38], Sect. 5.4.

Recall the notion of unnormalised Gaussians (Definition A.10). Due to the locality property, we can write

$$\tilde{t}_i(u_i) = N^U(u_i|b_i, \pi_i), \quad \boldsymbol{b} = (b_i)_i, \ \boldsymbol{\Pi} = \mathrm{diag}(\pi_i)_i,$$

where $\boldsymbol{b}$ and $\boldsymbol{\Pi}$ are the site parameters, thus each site approximation is an unnormalised one-dimensional Gaussian. If $\pi_i = 0$, we always set $b_i = 0$. For reasons of numerical stability, it is sensible to require that if $\pi_i \neq 0$, then it is bounded away from zero by some threshold. If $\pi_i \neq 0$, we also define $m_i = b_i/\pi_i$, the mean of $\tilde{t}_i(u_i)$ taken as a Gaussian. If $\pi_i = 0$, we set $m_i = 0$. Note that the EP posterior approximation $Q(\boldsymbol{u})$ depends only on $2\,n$ parameters in addition to the kernel matrix. In fact, we have

$$Q(\boldsymbol{u}) = N(\boldsymbol{u}|\boldsymbol{h}, \boldsymbol{A}), \quad \boldsymbol{A} = \left(\boldsymbol{K}^{-1} + \boldsymbol{\Pi}\right)^{-1}, \ \boldsymbol{h} = \boldsymbol{A}\boldsymbol{b} \tag{4.6}$$

[6] As an aside, note that this setting does not only include GP models. As has been observed in [41], it also represents certain *independent component analysis (ICA)* models in the presence of Gaussian noise on the observed (mixed) signals. In this case, the u_i would correspond to the latent sources, and the covariance of $P(\boldsymbol{u})$ is a function of the inverse mixing matrix and the noise variances.

(using Lemma A.11). The algorithm starts with $\boldsymbol{b} = \boldsymbol{0}$, $\boldsymbol{\Pi} = \boldsymbol{0}$, i.e. $\boldsymbol{h} = \boldsymbol{0}$, $\boldsymbol{A} = \boldsymbol{K}$, $Q(\boldsymbol{u}) = P(\boldsymbol{u})$, then iterates EP updates (Definition C.2) until convergence. In order to do an EP update, we require the marginal $Q(u_i) = N(h_i, a_{i,i})$. Details of how to do the update in the Gaussian case, based on the log partition function $\log Z_i = \log \mathrm{E}_{Q^{\setminus i}}[P(y_i|u_i)]$, are given in Appendix C.1.2. By the locality property, the update will only change the site parameters at i.

From the equations in Appendix C.1.2, we see that if $\log P(y_i|u_i)$ is concave in u_i for all possible y_i, then the tilted marginal $\hat{P}(u_i)$ always has positive variance. The new value of π_i has the same sign as the second derivative of $-\log Z_i$ w.r.t. $h_i^{\setminus i}$ (the mean of the cavity marginal $Q^{\setminus i}(u_i)$), thus is guaranteed to be positive if the log partition function is concave. Note that if the update results in $\pi_i < 0$, the marginal posterior approximation $Q(u_i)$ exhibits an increase in variance. Finally, after b_i and π_i have been updated, we have to adjust $\boldsymbol{h}$ and $\boldsymbol{A}$ (or at least $\operatorname{diag} \boldsymbol{A}$) as well. These updates dominate the cost for running EP, and since in the applications we are interested in $\boldsymbol{K}^{-1}$ does not follow any sparsity pattern the burden cannot be eased without further approximations. Since only one element in $\boldsymbol{\Pi}$ has changed, the updates can be done in $O(n^2)$ using the Woodbury formula (Lemma A.2). In practice, we would recommend to improve the condition of the kernel matrix $\boldsymbol{K}$ by adding a small multiple of the identity[7] (see [132]). In light of the definition of $\boldsymbol{A}$ in (4.6), it is also recommended for reasons of stability to restrict the components of $\boldsymbol{\Pi}$ to be nonnegative. In the context of a different approximation problem, Minka [125] reports that such restrictions can lead to worse results, but for GP binary classification we have not even encountered the case of a negative π_i so far.

EP for Gaussian process models has been proposed in [125, 124]. In fact, it is not hard to see that fixed points of EP in this case coincide with such for the *cavity TAP* approximation suggested earlier by Opper and Winther [139]. While the latter authors derived a set of self-consistent equations from a statistical physics ansatz which was then solved by iteration, EP provides a simpler, typically somewhat faster and more stable algorithm for this purpose. However, the algorithm

[7] Note that the problem of ill-conditioned $\boldsymbol{K}$ does not plague the *sparse* GP approximations to be developed below.

has time complexity $O(n^3)$ and requires $O(n^2)$ of memory, which is prohibitive for large training set sizes n. As we will see, it nevertheless provides the basis for the much more efficient sparse algorithms to be developed below.

4.4 Sparse Gaussian Process Methods: Conditional Inference

In this section, we discuss two schemes for sparse approximate Bayesian inference on GP models, conditional on a fixed set of hyperparameters. Both can be applied in principle to arbitrary likelihoods, although we concentrate here on the case of completely factorised likelihoods, in which the sites are arbitrary positive functions of one component in the latent vector $\boldsymbol{u}$ each. The *informative vector machine (IVM)* is the simpler of the both, and the basic scheme can be seen as a simplified version of the sparse on-line method of [40]. However, the IVM is a greedy algorithm, thus algorithmically quite different from an on-line scheme, and its key merits as compared to other sparse methods are high efficiency and simplicity of implementation. An early version has been proposed in [97], while the fully developed scheme (as presented here) appears in [98]. The *projected latent variables (PLV)* algorithm is a more expensive and potentially more accurate scheme. It can be seen as greedy version of the batch algorithm presented in [38, 41], where instead of randomly running over the training data and including new points in an on-line fashion, we try to obtain a more efficient scheme by ordering candidates in a heuristic manner. This is quite a bit more challenging than for the simpler IVM, but the more accurate likelihood approximation employed in PLV may render the method more suitable for model selection (see Section 4.5.1).

4.4.1 The Informative Vector Machine

In a nutshell, the *informative vector machine (IVM)* employs a likelihood approximation (see Section 4.2.1) which follows from using EP (see Section 4.3) but

leaving most of the site parameters clamped at zero. The patterns for which ADF updates are done, are chosen using greedy forward selection based on differential criteria (see Section 4.2.2). The chief merits of the method are its simplicity and high efficiency.

As mentioned in Section 4.2.1, a simple way of obtaining a likelihood approximation in the context of the EP approximation for GP models is to clamp most of the site parameters to zero, i.e. $\pi_i = 0$, $b_i = 0$ for all $i \notin I$. Here, $I \subset \{1, \ldots, n\}$ will be called the *active set*[8], $|I| =: d < n$. As discussed in Section 4.2.2, we can use greedy forward selection to select new patterns to be included into I: compute a score function for all points in a *selection index* $J \subset \{1, \ldots, n\} \setminus I$ and pick the winner.

What representation do we have to maintain and update in order to run the IVM scheme? It is easy to see that if I was chosen purely at random, a representation of $Q(\boldsymbol{u})$ sufficient for these training updates and for future predictions would require $O(d^2)$ parameters only, and the training algorithm would run in time depending on d only. However, such random selection strategies can perform very poorly especially for the case of difficult binary classification or regression estimation tasks and $d \ll n$. For a greedy selection strategy, it seems obvious that in order to make any statement about the potential value of an inclusion of point i, we at least need to know the current marginal $Q(u_i)$. Although a minimal $O(d^2)$ representation can provide this marginal on demand, the cost for the evaluation is $O(d^2)$, which is prohibitive if selections are done from a large set J of candidates. A representation which allows the efficient update of the marginals $Q(u_i)$ for a large number of points $i \notin I$, and which can be updated in a numerically stable way after inclusions, is described in Appendix C.3.1. The storage requirements are dominated by one $d \times n$ matrix. For a fixed active set I, the posterior approximation $Q(\boldsymbol{u}_I)$ is $N(\boldsymbol{K}_I \boldsymbol{\Pi}_I^{1/2} \boldsymbol{L}^{-T} \boldsymbol{\beta}, \boldsymbol{K}_I \boldsymbol{\Pi}_I^{1/2} \boldsymbol{B}^{-1} \boldsymbol{\Pi}_I^{-1/2})$, with

$$\boldsymbol{\beta} = \boldsymbol{L}^{-1} \boldsymbol{\Pi}_I^{-1/2} \boldsymbol{b}_I \tag{4.7}$$

and $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T$ as defined in (C.4). Recall that $\boldsymbol{b}_I$, $\boldsymbol{\Pi}_I$ are the parameters of the

[8] I should not be confused with active sets as defined in the context of constrained optimisation problems (see [56]). While the latter can be found in polynomial time for convex problems, the task of choosing I in some "optimal" way is typically of combinatorial complexity.

active site approximations. The predictive distribution $Q(u_*|\boldsymbol{x}_*, S)$ is obtained by averaging $P(u_*|\boldsymbol{u}_I)$ over $Q(\boldsymbol{u}_I)$, resulting in a Gaussian process with mean and variance

$$\begin{aligned} \mu(\boldsymbol{x}_*) = \boldsymbol{\beta}^T \boldsymbol{m}_*, \quad \sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \|\boldsymbol{m}_*\|^2, \\ \boldsymbol{m}_* = \boldsymbol{L}^{-1} \boldsymbol{\Pi}_I^{1/2} \boldsymbol{k}_I(\boldsymbol{x}_*), \end{aligned} \tag{4.8}$$

where $\boldsymbol{k}_I(\boldsymbol{x}_*) = (K(\boldsymbol{x}_*, \boldsymbol{x}_i))_{i \in I}$ (see Equation 2.14). If the predictive variance is not required, it is sensible to pre-compute the prediction vector $\boldsymbol{\xi} = \boldsymbol{\Pi}_I^{1/2} \boldsymbol{L}^{-T} \boldsymbol{\beta}$, then $\mu(\boldsymbol{x}_*) = \boldsymbol{\xi}^T \boldsymbol{k}_I(\boldsymbol{x}_*)$. Finally, the predictive target distribution $Q(y_*|\boldsymbol{x}_*, S)$ is obtained by averaging $P(y_*|u_*)$ over $Q(u_*|\boldsymbol{x}_*, S)$. For binary classification and probit noise $P(y_*|u_*)$ (2.9), we have

$$Q(y_*|\boldsymbol{x}_*, S) = \Phi\left(\frac{y_*(\mu(\boldsymbol{x}_*) + b)}{\sqrt{1 + \sigma^2(\boldsymbol{x}_*)}}\right), \tag{4.9}$$

where b is a bias hyperparameter. For other noise models $P(y_*|u_*)$, the one-dimensional integral can be approximated by Gaussian quadrature (see Section C.1.2). In fact, for commonly used noise models, the Bayes classifier is $\mathrm{sgn}(\mu(\boldsymbol{x}_*)+b)$, independent of $\sigma^2(\boldsymbol{x}_*)$. This follows from the fact that $Q(u_*|\boldsymbol{x}_*, S)$ is symmetric around $\mu(\boldsymbol{x}_*)$. If we have the option of rejecting a fraction of test points, the decision should be based on the size of $|Q(y_*|\boldsymbol{x}_*, S) - 1/2|$. More generally, if an uneven loss function (in the decision-theoretic sense) is appropriate for our task, our decision should be the one minimising the posterior risk (i.e. expected loss).

An important fact which distinguishes the IVM likelihood approximation from other more expensive ones, is that knowledge of the marginal $Q(u_i)$ is also sufficient for evaluating a range of sensible selection scores at i in $O(1)$: a direct consequence of the locality property of EP on models with factorised likelihood (Lemma C.1). Recall the differential entropy score and the information gain score from Section 4.2.2. Due to the locality property, the former is simply the difference between the *marginal* entropies before and after inclusion (see Lemma C.1), i.e.

$$\Delta_i^{\mathrm{Ent}} = \mathrm{H}[Q^{new}(u_i)] - \mathrm{H}[Q(u_i)] = \frac{1}{2} \log \frac{\lambda_i^{new}}{a_{i,i}} = \frac{1}{2} \log(1 - \lambda_i \nu_i) + \frac{1}{2} \log \frac{\lambda_i}{a_{i,i}}, \tag{4.10}$$

where $Q(u_i) = N(h_i, a_{i,i})$ (recall the notation and definitions from Section 4.3 and Appendix C.1.2). Given that the marginal is known, this term can be computed in $O(1)$. Note that Δ_i^{Ent} is proportional to the reduction in log variance from $Q(u_i)$ to $Q^{new}(u_i)$. The information gain score can be computed equally easily, by again exploiting Lemma C.1:

$$\Delta_i^{\mathrm{Info}} = -\mathrm{D}[Q^{new}(\boldsymbol{u}) \,\|\, Q(\boldsymbol{u})] = -\mathrm{D}[Q^{new}(u_i) \,\|\, Q(u_i)].$$

Using Equation A.17 we have

$$\Delta_i^{\mathrm{Info}} = -\mathrm{D}[Q^{new}(u_i) \,\|\, Q(u_i)] = \frac{1}{2}\left(\log\frac{\lambda_i^{new}}{a_{i,i}} - \frac{\lambda_i^{new}}{a_{i,i}} - \alpha_i^2 a_{i,i} - 1\right), \qquad (4.11)$$

where we have used that $h_i^{new} - h_i = \alpha_i a_{i,i}$ (see Appendix C.1.2). Again, this is $O(1)$, given the marginal moments h_i, $a_{i,i}$. From (4.8), we see that the computation of $Q(u_i)$ hinges on knowledge of column i of matrix $\boldsymbol{M} = \boldsymbol{L}^{-1}\boldsymbol{\Pi}_I^{1/2}\boldsymbol{K}_{I,\cdot}$ (Equation C.5), the so-called "stub" for i. This completes the description of the IVM scheme (Algorithm 1).

Algorithm 1 Basic informative vector machine algorithm

Require: Desired sparsity $d \ll n$, threshold $\varepsilon > 0$ (numerical stability).

$I = \emptyset$, $\boldsymbol{b} = \boldsymbol{0}$, $\boldsymbol{\Pi} = \mathrm{diag}(\boldsymbol{0})$, $\mathrm{diag}\,\boldsymbol{A} = \mathrm{diag}\,\boldsymbol{K}$, $\boldsymbol{h} = \boldsymbol{0}$, $J = \{1, \ldots, n\}$.

repeat

  for $j \in J$ **do**

    Compute $\Delta_j = \Delta_j^{\mathrm{Ent}}$ (Equation 4.10) (or $\Delta_j = \Delta_j^{\mathrm{Info}}$ (Equation 4.11)).

  end for

  $i = \mathrm{argmax}_{j \in J}\{\Delta_j \,|\, \pi_j^{new} > \varepsilon\}$

  Here, π_j^{new}, b_j^{new} are computed by ADF update (see Appendix C.1.2).

  Site updates: $\pi_i \leftarrow \pi_i^{new}$, $b_i \leftarrow b_i^{new}$.

  Update representation as shown in Appendix C.3.1, notably $\boldsymbol{L}$, $\boldsymbol{M}$, $\mathrm{diag}\,\boldsymbol{A}$, $\boldsymbol{h}$.

  $I \leftarrow I \cup \{i\}$, $J \leftarrow J \setminus \{i\}$.

until $|I| = d$

The representation described in Appendix C.3.1 allows us to keep all n marginals $Q(u_i)$ up-to-date at any time during the algorithm, at a cost of $O(n\,d)$ per inclu-

sion, leading to a total cost of $O(n\,d^2)$ with a small constant. In this setup, in order to decide which point to include next, we pick the top-scorer among *all* points outside of I. This strategy seems wasteful, especially during later iterations when the impact of new inclusions on Q diminishes. In a *randomised greedy* version of the algorithm, the selection index J is a strict subset of $\{1,\ldots,n\} \setminus I$ during later iterations, and only the marginals corresponding to these points have to be available for scoring. We refer to this procedure as *randomised greedy selection (RGS)*. In Appendix C.3.1, we describe an updating scheme for the representation of Q which exploits this selection constraint by delaying the computation of stubs as long as their patterns are not in the selection index. The more inertia the selection index J exhibits between iterations, the cheaper the computational cost becomes. This can be supported by retaining a fraction $\tau \in (0,1)$ of the top-scorers from the previous iteration in J, filling it up with new points drawn at random. Our options range from "conservative" (large J, small τ) to "risky" (rather small J, large τ). During early iterations, it seems sensible to use full greedy selections, not only because early inclusions typically have a larger impact on the final Q than later ones, but also because a complete update of all marginals is much cheaper while the active set is still small. Our present implementation allows memory usage to be limited to $O(n\,d_{lim})$ with d_{lim} smaller than the final active set size. For very large n, it would become necessary to add refinements like a cache hierarchy, see Section C.3.1 for some ideas.

Finally, we may want to address the problem of how many points d to include into the final active set. First of all, one of the important advantages of the IVM over other "sparse" methods such as SVM (see Section 2.1.6) is that d can be controlled. The algorithm will always remain within given resource constraints, and its running time is predictable *a priori*. Apart from this, a number of heuristics may be used to select an appropriate d from a given resource-constrained range. For example, the error on an independent validation set can be monitored using predictions via (4.8), requiring $O(m\,d)$ if m is the number of validation points (one would not have to compute the error after each inclusion). Another idea is to monitor the error or predictive log likelihood on the remaining training sam-

ple $S_{\setminus I}$ outside the current active set. This typically underestimates true error, because the selection criteria prefer patterns which are wrongly classified by the current predictor, but it may nevertheless be useful for deciding when to stop. Evaluating such stopping heuristics empirically is subject to future work.

In the IVM scheme, the decision to include i into I is final. It is easy to modify the scheme to allow for "exchange moves": for $i \in I$, $j \notin I$, such a move exchanges j against i in I, using an EP update step to modify the site parameters. In Appendix C.3.2, we give details about exchange moves, showing how the representation is updated and how the differential scores can be modified to score "exchange pairs" (i, j). Exploring the usefulness of exchange moves in practice is not in the scope of this thesis and subject to future work. Potential difficulties include numerical instabilities when updating the representation and the much larger number of exchange candidate *pairs* to be scored in every iteration.

Finally, the reader may wonder why we do not run EP updates until convergence once the final active set I has been chosen. Indeed, we did so in preliminary experiments, but frequently noted a decline in performance after the additional EP update iterations. We do not have a conclusive explanation for this effect in the moment, but can merely offer the plausible argument that subsequent EP iterations will indeed improve the approximation to the posterior $P(\boldsymbol{u}|S_I)$ for S_I only, thereby worsening the approximation to the full posterior $P(\boldsymbol{u}|S)$.

4.4.1.1 Running Time Complexity

One attractive feature of the IVM scheme, in contrast to other well-known kernel methods such as SVM is that the running time requirements can be estimated and controlled *a priori*. The IVM scheme without RGS requires $O(n\,d')$ for each inclusion with current active set size d', thus $O(n\,d^2)$ in total. The dominating operation during an inclusion is a matrix-vector multiplication with a matrix of size $n \times d'$. For RGS, the dominating operations are "stub updates": it needs $O(d')$ in order to extend a stub from size d' to $d'+1$, so the cost per inclusion is at least $O(|J|\,d')$ where $|J|$ is the current selection index. The exact cost depends on the details of the setup, notably the retain fraction τ and the cache mechanism

for the stubs, but is well below $O(n\,d')$ if $|J| \ll n$.

In practical applications involving numerical mathematics, much can be gained by organising computations such that the dominant inner loops consists of simple vector operations such as $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x}^T \boldsymbol{y}$ or $(\boldsymbol{x}, \boldsymbol{y}) \mapsto \boldsymbol{x} + \alpha \boldsymbol{y}$, with as large vectors as possible. Here lies another advantage of IVM over schemes such as SVM trained with the popular SMO algorithm [142]. The dominating computation for IVM without RGS consists of a single $O(n\,d')$ matrix-vector multiplication[9] per inclusion, while SMO is typically dominated by a very large number of elementary operations, or even covariance function evaluations, which cannot be vectorised efficiently. If IVM is run with RGS and $|J| \ll n$, not much larger than d, the size of vectorisation is decreased. Individual stub updates require inner products of vectors of size d'. If $|J| \gg d$, it makes sense to bundle these to attain vectorisation of size $|J|$, but this is not supported by our implementation in the moment.

High efficiency becomes crucial when the conditional inference described here is used as subroutine for model selection (see Section 4.5.2). Measurements show that often the time requirements for conditional inference dominate the total time for computing the model selection criterion and its gradient, especially if RGS is involved. One reason for this is that the remaining dominating computations for the gradient can be vectorised efficiently. Efforts to speed up conditional inference should concentrate on reducing the organisatorial overhead for the stub cache, or to bundle stub updates in some way to increase the vectorisation size. Our implementation includes one feature worth mentioning. Recall that for small initial active set sizes d', the selection index is $J = \{1, \ldots, n\}$. To obtain efficient vectorisation, we require the rows of $\boldsymbol{M}$ to be accessible as linear vectors. Later on during RGS, columns of $\boldsymbol{M}$ need to be accessed as linear vectors to do efficient stub updates. Therefore, once RGS is activated the principal buffer matrix (storing the full $\boldsymbol{M}$ initially, and being fed by the stub cache later) has to be transposed. Since this matrix can be large, it is sensible (or even essential) to do this transposition in place, *i.e.* without requiring a second large buffer. In place

[9] This is a BLAS-2 operation [51], for which the BLAS suite provides highly efficient implementations for many architectures. Vector-vector operations are supported via BLAS-1.

transposition of non-square matrices is non-trivial, but efficient algorithms exist [26]. Certainly, other speed-up techniques could be applied to our code[10], but we leave this for future work.

4.4.2 Projected Latent Variables

In contrast to the IVM which employs a factorised approximation to the likelihood $P(S|\boldsymbol{u})$, the PLV scheme uses an approximation in which each latent variable u_i, $i \in I$ is coupled with all training datapoints, based on the so-called KL-optimal projection motivated by Lemma 4.1. If $P(S|\boldsymbol{u})$ itself is an unnormalised Gaussian w.r.t. $\boldsymbol{u}$, say $\propto N^U(\boldsymbol{u}|\boldsymbol{b}, \boldsymbol{\Pi})$, the corresponding likelihood approximation is $N^U(\mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I] \,|\, \boldsymbol{b}, \boldsymbol{\Pi})$, where $\mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I] = \boldsymbol{P}^T \boldsymbol{u}_I$ the prior conditional expectation (recall that $\boldsymbol{P} = \boldsymbol{K}_I^{-1} \boldsymbol{K}_{I,\cdot}$). If $P(S|\boldsymbol{u})$ is not Gaussian, we can embed the mapping via $\boldsymbol{P}^T$ into a scheme which renders Gaussian posterior approximations (such as EP), since $\boldsymbol{P}$ does not depend on the training targets. Apart from its "KL-optimality", the approximation can intuitively be understood as a modification of the sampling model for the targets $\boldsymbol{y}$. Namely, we draw $\boldsymbol{u}_I \sim P(\boldsymbol{u}_I)$, $\boldsymbol{u}_{\setminus I} \sim P(\boldsymbol{u}_{\setminus I}|\boldsymbol{u}_I)$, but instead of sampling $\boldsymbol{y} \sim P(\boldsymbol{y}|\boldsymbol{u})$, we replace $\boldsymbol{u}_{\setminus I}$ in this conditional distribution by the mean of the distribution $P(\boldsymbol{u}_{\setminus I}|\boldsymbol{u}_I)$ it has been sampled from. We have seen in Section 4.2.1 that in order to obtain a sparse approximation, we have to constrain the likelihood to depend on $\boldsymbol{u}_I$ only, and the method of replacing the remaining variables $\boldsymbol{u}_{\setminus I}$ in the likelihood by their conditional mean[11] seems in a way least intrusive.

When compared with IVM, the PLV scheme is somewhat more complicated to develop and implement and its running time and memory requirements are larger. Furthermore, in the non-Gaussian likelihood case tensions might arise from the fact that the two principle underlying approximations (KL-optimal projections and EP) are "incompatible" in the sense discussed below, and finally suitable forward selection criteria are harder to compute and additional approximations will be required. Where is the potential gain of preferring PLV over IVM? From

[10] The present version does not incorporate highly efficient numerical packages such as BLAS [51] or LINPACK, but this will be done in the future.

[11] The best linear predictor (in the least-squares sense).

the experiments we have done to compare the methods (Sections 4.8.2 and 4.8.3) we can see that PLV can work successfully (especially in conjunction with automatic model selection) with smaller active set sizes d. An intuitive explanation is that due to the coupling in the PLV likelihood approximation points in the active set can summarise the information of neighbouring points more accurately. In IVM, points outside the active set influence predictions only in the weak sense of shaping the selection of I, while in PLV every point takes influence on predictions directly (the ones in I do so most strongly).

Due to the more elaborate likelihood approximation, representations and updates are somewhat more complicated for PLV than for the IVM, and the final algorithm is more expensive to run. However, very significant simplification can be obtained for the special case of GP regression with Gaussian noise (see Section 2.1.2). In the latter special case, we can fix $\boldsymbol{b} = \sigma^{-2}\boldsymbol{y}$ and $\boldsymbol{\Pi} = \sigma^{-2}\boldsymbol{I}$, while in general these site parameters have to be adjusted iteratively using EP updates. We have to address the following issues:

1. What representation allows stable and efficient updates, both to include new points into I and to update site parameters? How to do predictions based on this representation?

2. How to update the site parameters b_i, π_i of a point i which may or may not be in I? How to include a point i into the active set I?

3. How to score a point i w.r.t. the potential value of including it into I?

Before delving into the details, it is helpful to state the key differences to the simpler IVM scheme which will serve to motivate the differences in representation and updates. First, in PLV every u_i, $i \in I$ is coupled with all n sites. This means that each time a new index i is included into I, there is an incentive to update *all* $O(n)$ site parameters. Second, the coupling is via the matrix $\boldsymbol{P} = \boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,\cdot}$, namely $\boldsymbol{\gamma} = \mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I] = \boldsymbol{P}^T\boldsymbol{u}_I$. The coefficients γ_i are called *pseudo-variables* (or *projected latent variables*) and fulfil roughly the same role as the u_i themselves in the IVM scheme. Once $\boldsymbol{u}$ is replaced by $\boldsymbol{\gamma}$, the mechanics of PLV become quite similar to that of IVM. For example, in order to update the i-th

site approximation, we will require the marginal $Q(\gamma_i)$ (instead of $Q(u_i)$ in IVM). Some form of $\boldsymbol{P}$ will have to be maintained explicitly in order to map between $\boldsymbol{\gamma}$ and $\boldsymbol{u}_I$. Note that $\boldsymbol{\gamma}_I = \boldsymbol{u}_I$. ADF updates in PLV are done in the same way as in IVM, but using pseudo-variables γ_i instead of u_i and "blurred sites"

$$\tilde{t}_i(\gamma_i) = \int t_i(u_i) P(u_i|\boldsymbol{u}_I)\, du_i = \int t_i(u_i) N(u_i|\gamma_i, l_i^2)\, du_i,$$

where $l_i^2 = \boldsymbol{K}_{i,i} - p_i = \boldsymbol{K}_{i,i} - \boldsymbol{K}_{i,I}\boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,i}$ (p_i is introduced below).

Again, a suitable representation which allows for efficient and stable updates, is at the core of the algorithm. We have already noted that a form of $\boldsymbol{P}$ has to be stored. Please note that overlaps with the notation used for the IVM are unavoidable for reasons of simplicity. Thus, whatever variables are defined for the inner representations of IVM or PLV, these are meant to be *local* definitions and are not to be transferred to other contexts. For fixed I, let $\boldsymbol{K}_I = \boldsymbol{L}\boldsymbol{L}^T$ be decomposed into Cholesky factors, and let

$$\boldsymbol{V} = \boldsymbol{L}^{-1}\boldsymbol{K}_{I,\cdot}, \quad \boldsymbol{M} = \boldsymbol{I} + \boldsymbol{V}\boldsymbol{\Pi}\boldsymbol{V}^T.$$

Note that $\boldsymbol{P} = \boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,\cdot} = \boldsymbol{L}^{-T}\boldsymbol{V}$, thus $\boldsymbol{V}$ is nothing but a convenient representation of the projection matrix. In practice, we would recommend using a *jitter factor* in the covariance function (see [132]), which leads to $\boldsymbol{K}$ being replaced by $\boldsymbol{K} + \tau\boldsymbol{I}$ for a small $\tau > 0$. This makes sure that the Cholesky factor $\boldsymbol{L}$ can be computed without problems. Also $\boldsymbol{\gamma} = \boldsymbol{V}^T\boldsymbol{L}^{-1}\boldsymbol{u}_I$, so that

$$Q(\boldsymbol{u}) \propto P(\boldsymbol{u}) N^U(\boldsymbol{b}|\boldsymbol{\gamma}, \boldsymbol{\Pi}). \tag{4.12}$$

It is easy to see that $Q(\boldsymbol{u}_I) = N(\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{V}\boldsymbol{b}, \boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T)$. Let $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{Q}^T$ be decomposed into Cholesky factors. Also, define $\boldsymbol{\beta} = \boldsymbol{Q}^{-1}\boldsymbol{V}\boldsymbol{b}$. Then, the posterior can be written as

$$Q(\boldsymbol{u}_I) = N\left(\boldsymbol{u}_I \,|\, \boldsymbol{L}\boldsymbol{Q}^{-T}\boldsymbol{\beta}, \boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T\right).$$

The predictive posterior is now obtained from (2.14). Namely, for a new datapoint $\boldsymbol{x}_*$ the predictive mean $\mu(\boldsymbol{x}_*)$ and variance $\sigma^2(\boldsymbol{x}_*)$ are

$$\begin{aligned} \mu(\boldsymbol{x}_*) = (\boldsymbol{l}_*^M)^T\boldsymbol{\beta}, \quad \sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \|\boldsymbol{l}_*\|^2 + \|\boldsymbol{l}_*^M\|^2, \\ \boldsymbol{l}_* = \boldsymbol{L}^{-1}\boldsymbol{k}_*, \quad \boldsymbol{l}_*^M = \boldsymbol{Q}^{-1}\boldsymbol{l}_*, \end{aligned} \tag{4.13}$$

where $\boldsymbol{k}_* = (K(\boldsymbol{x}_*, \boldsymbol{x}_i))_{i \in I}$. Thus, prediction is $O(d^2)$, yet requires two back-substitutions[12] instead of only one as for the IVM. Note that if the predictive mean is required only, one should pre-compute[13] the prediction vector $\boldsymbol{\xi} = \boldsymbol{L}^{-T}\boldsymbol{Q}^{-T}\boldsymbol{\beta}$, after which $\mu(\boldsymbol{x}_*) = \boldsymbol{k}(\boldsymbol{x}_*)^T\boldsymbol{\xi}$ is $O(d)$.

If the likelihood $P(y_i|u_i)$ is not Gaussian, we have to adjust the site parameters iteratively so as to obtain a good Gaussian approximation to the true posterior. In PLV, this is done by EP updates. Define

$$\boldsymbol{p} = \operatorname{diag} \boldsymbol{V}^T\boldsymbol{V}, \quad \boldsymbol{\mu} = \boldsymbol{V}^T\boldsymbol{Q}^{-T}\boldsymbol{\beta}, \quad \boldsymbol{q} = \operatorname{diag} \boldsymbol{V}^T\boldsymbol{M}^{-1}\boldsymbol{V}. \tag{4.14}$$

The update formulae are derived in Appendix C.4.1. We have already noted above that they are virtually identical to the updates for IVM once $\boldsymbol{u}$ is replaced against the pseudo-variable vector $\boldsymbol{\gamma}$ and the true site $t_i(u_i)$ against the "blurred site" $\tilde{t}_i(\gamma_i)$. Note the meaning of the components in (4.14): $P(u_i|\boldsymbol{u}_I) = Q(u_i|\boldsymbol{u}_I) = N(u_i|\gamma_i, l_i^2)$ with $l_i^2 = \boldsymbol{K}_{i,i} - p_i$, and $Q(\gamma_i) = N(\gamma_i|\mu_i, q_i)$. Note that while $\boldsymbol{p}$ is independent of the site parameters, $\boldsymbol{\mu}$ and $\boldsymbol{q}$ are not, and in PLV the components of the latter are computed on demand. A change in the site parameters will modify $\boldsymbol{M}$, so that $\boldsymbol{Q}$ and related variables have to be updated. Since $\boldsymbol{M}^{new} = \boldsymbol{M} + (\pi_i^{new} - \pi_i)\boldsymbol{v}_i\boldsymbol{v}_i^T$, where $\boldsymbol{v}_i = \boldsymbol{V}_{\cdot,i}$, this can be done using the technique described in Section A.2.2. Finally, in order to include a new point i into I, the representation has to be updated, Appendix C.4.1 contains the details. Immediately after the inclusion, the site parameters of i should be updated. In general, when updating site parameters after the inclusion of i, the ones of i itself should probably be visited several times.

Note that $\operatorname{Var}[\boldsymbol{\gamma}] = \boldsymbol{V}^T\boldsymbol{M}^{-1}\boldsymbol{V}$, measuring the covariance between pseudo-variables γ_j. Define $\boldsymbol{C} = \boldsymbol{Q}^{-1}\boldsymbol{V}$, then $\operatorname{Var}[\boldsymbol{\gamma}] = \boldsymbol{C}^T\boldsymbol{C}$. If $\boldsymbol{c}_i = \boldsymbol{C}_{\cdot,i}$, then $\mu_i = \boldsymbol{c}_i^T\boldsymbol{\beta}$, $q_i = \|\boldsymbol{c}_i\|^2$, so columns of $\boldsymbol{C}$ play an important role in updating and scoring (they are the equivalents of "stubs" in IVM). However, as opposed to $\boldsymbol{V}$,

[12] For maximum efficiency, this can be reduced to a single Cholesky factor after training. The most numerically stable way of doing this uses the singular value decomposition (SVD) of $\boldsymbol{M}$, possibly treating eigenvalues very close to 1 in a special way.

[13] In the case of full GP inference, Gibbs [63] reports stability problems with this pre-computation approach. Again, we would expect that these problems are alleviated in the case of sparse approximations, especially if a jitter factor is used for $\boldsymbol{K}_I$.

$\boldsymbol{C}$ cannot be maintained and updated incrementally, its columns are computed on demand.

Note that if we were to update all site parameters after each inclusion of a new point into I, this would cost $O(n\,d^2)$ per inclusion, thus leading to a prohibitive overall cost of $O(n\,d^3)$. Csató [38] circumvents this problem by choosing an on-line like scheme, cycling at random over the data and deciding individually based on some threshold whether to include a point or merely update its site parameters. In our scheme, site parameter updates are separated from inclusion decisions, the latter are based on greedy selection. After each inclusion, a set of points L, $I \subset L$, is picked at random and only their site parameters are updated. The details can be found in Appendix C.4.1, the cost is $O(d^2)$ for each update, so $O(|L|\,d^2)$ for a sweep over all sites in L.

Finally note the subtle difference in how the scheme is run if the site parameters are fixed from the beginning (as in GP regression with Gaussian noise). If we were to update the site parameters for some $i \notin I$ in the same way as in the general case, they would become *different* from their fixed values even though the true likelihood is Gaussian. This is because the KL-optimal projection to obtain the likelihood approximation is not perfectly compatible with the EP scheme.[14] Another consequence is that if we adapted site parameters using EP in this setting, the final Q would be a worse approximation to the true (Gaussian) posterior in the sense of KL-optimality (due to Lemma 4.1) than if we fixed them (which is what we do).

4.4.2.1 Scoring Points for Inclusion

Can the selection heuristics discussed in Section 4.2.2 be applied for PLV as well? For the IVM, these differential criteria turned out to depend on the old and new *marginal* distributions at the point in question only, a consequence of the locality property of EP with a factorised likelihood. In PLV, the likelihood (approximation) is *not* factorised anymore, all sites depend on all of $\boldsymbol{u}_I$ and are

[14] Recall from Section 4.2.1 that the KL-optimal approximation uses an e-projection (see Section A.4.2), while EP uses m-projections into the Gaussian family.

therefore coupled. In order to score a point with one of the criteria, we have to spend the same time as would be required to include the point (which is dismissed as infeasible in Section 4.2.2).

We can make progress nevertheless by considering approximations to the criteria we used for the IVM. The problem is that once i is included, u_i is coupled with all sites immediately. A simple approximation would be to ignore all these couplings except with site i. To be specific, let $S(Q, Q^{new})$ be the full criterion, where Q is the current approximate posterior and Q^{new} is obtained from Q by inclusion of i. $S(Q, Q^{new})$ cannot be computed faster than updating the representation for $Q \to Q^{new}$. The cavity distribution $Q^{\backslash i}$ is obtained from Q by removing site i together with the pseudo-variable $\gamma_i = (\boldsymbol{P}^T \boldsymbol{u}_I)_i$. Define $\tilde{Q}(\boldsymbol{u})$, an approximation to $Q^{new}(\boldsymbol{u})$ as

$$\tilde{Q}(\boldsymbol{u}) \propto Q^{\backslash i}(\boldsymbol{u}) N^U(u_i | \tilde{b}_i, \tilde{\pi}_i), \tag{4.15}$$

where $\tilde{b}_i$, $\tilde{\pi}_i$ are obtained by an ADF update with i added to I. As is detailed in Section C.4.2, these are in general different from b_i, π_i even if the latter have just been updated. Furthermore, $\tilde{S} = S(Q, \tilde{Q})$.[15] In contrast to S, $\tilde{S}$ can be computed in $O(1)$ given our representation, as is shown in Appendix C.4.2 for the special case of the information gain criterion (the derivations for other criteria should be similar). In fact, $\tilde{S}$ may be a rather coarse approximation to S in many cases, however the vastly reduced cost is a strong argument to explore its usefulness more thoroughly.

There are intermediates between $\tilde{S}$ and S. For example, we can choose a small index set H with $H \cap I' = \emptyset$, where again $I' = I \cup \{i\}$, set $H' = H \cup \{i\}$ and use

$$\begin{aligned} \tilde{Q}(\boldsymbol{u}) &\propto Q^{\backslash H'}(\boldsymbol{u}) N^U(\tilde{\boldsymbol{b}}_{H'} \,|\, (\boldsymbol{P}'_{\cdot,H'})^T \boldsymbol{u}_{I'}, \tilde{\boldsymbol{\Pi}}_{H'}), \\ Q^{\backslash H'}(\boldsymbol{u}) &\propto P(\boldsymbol{u}) N^U\left(\boldsymbol{b}_{\backslash H'} \,|\, (\boldsymbol{P}_{\cdot,\backslash H'})^T \boldsymbol{u}_I, \boldsymbol{\Pi}_{\backslash H'}\right). \end{aligned} \tag{4.16}$$

Here, $\boldsymbol{P}' \in \mathbb{R}^{d+1,n}$ denotes the projection matrix *after* inclusion of i, and $\tilde{\boldsymbol{b}}_{H'}$, $\tilde{\boldsymbol{\Pi}}_{H'}$ may be different from the old values $\boldsymbol{b}_{H'}$, $\boldsymbol{\Pi}_{H'}$ defining Q. For small H, this

[15] It seems that we have ignored couplings here between site i and the $\boldsymbol{u}_I$. However, note that once i is included into I and the likelihood approximation depends on u_i, there is no (direct) coupling between these variables, because $\boldsymbol{P}_{\cdot,I} = \boldsymbol{I}$.

criterion can still be computed feasibly, as is shown in Appendix C.4.3. Note that the full information gain criterion is obtained for $H = \{1, \ldots, n\} \setminus I'$, and the coupling-free one for $H = \emptyset$. Of course, this raises new questions, for example how to choose H or whether and how to update the parameters $\boldsymbol{b}_{H'}$, $\boldsymbol{\Pi}_{H'}$ under the new coupling before evaluating the score. Some ideas towards the former are discussed below. It seems sensible to update the site parameters for H' using EP steps before evaluating the criterion, otherwise they may be poorly adapted to the new situation and the increased coupling may not pay off at all. This is not hard to do, the details are developed in Appendix C.4.3. If $|H'| \leq d$, the dominating computation for these updates and the score computation is the evaluation of $\boldsymbol{C}_{\cdot,H'}$ (we need the covariance of $\boldsymbol{\gamma}_{H'}$ under Q), which is $O(|H'|\, d^2)$. This holds even if $|H'| > d$. We also require the elements $\boldsymbol{K}_{i,H'}$ of the kernel matrix and $|H'|$ elements of a new row of $\boldsymbol{V}$. If $|H'| > d$, the cost of $O(|H'|\, d^2)$ soon becomes prohibitive. However, note that for the special case $\tilde{\boldsymbol{b}}_{H'} = \boldsymbol{b}_{H'}$, $\tilde{\boldsymbol{\Pi}}_{H'} = \boldsymbol{\Pi}_{H'}$ *without* updating, the cost of evaluating the criterion drops to $O(|H'|\, d)$.

When using the simple coupling-free approximation, it is more efficient to interleave score computations and site parameter updates. After the inclusion of a new point, we first update the site parameters for the new active set I together with some randomly chosen points from J. However, in the subsequent systematic update run over J we compute the score for each pattern immediately after the update of its site parameters. This is slightly problematic since we are now comparing score values based on different site parameter settings, but it leads to a significant gain in efficiency.

4.4.2.2 Overview of the Algorithm

In this section, we give an overview of the algorithm if the simple coupling-free score approximation is used (see Section 4.4.2.1). The algorithm consists of two stages. In the first stage, the active set I is chosen by greedy forward selection until a desired size d is attained. In the second stage, the site parameters $\boldsymbol{b}$, $\boldsymbol{\Pi}$ are thoroughly updated (within resource constraints). The first stage consists of iterations each of which leads to the inclusion of a new pattern into I. It is

initialised by picking a few points at random for an initial I and setting $\boldsymbol{b} = \mathbf{0}$, $\boldsymbol{\Pi} = \operatorname{diag} \mathbf{0}$. Algorithm 2 shows the schema for one iteration.

Algorithm 2 PLV: First-stage iteration, leading to the inclusion of one pattern.

Require: Representation for current active set I up-to-date.

if Site parameters not fixed **then**
  Pick $L \subset \{1, \ldots, n\}$, $I \subset L$.
  for $j \in L$ **do**
    Update site parameters for j (see Appendix C.4.1).
  end for
  Pick $J \subset \{1, \ldots, n\}$, $J \cap I = \emptyset$.
  for $j \in J$ **do**
    Update site parameters for j. Compute coupling-free score approximation $\Delta_j = S(Q, \tilde{Q})$.
  end for
else
  Pick $J \subset \{1, \ldots, n\}$, $J \cap I = \emptyset$.
  for $j \in J$ **do**
    Compute coupling-free score approximation $\Delta_j = S(Q, \tilde{Q})$.
  end for
end if
$i = \operatorname{argmax}_{j \in J} \Delta_j$
Update representation as shown in Appendix C.4.1.
$I \leftarrow I \cup \{i\}$

The initial update run over L is required because the site parameters have not been updated since the last recent inclusion into I. During the run over L, patterns from I (and especially the one included last recently) should probably be visited several times. If the resource limits permit it, we would recommend $L = I \cup J$. Note that score computations and site updates are interleaved. For reasons of stability, we update the site parameters for i immediately after its inclusion into I, and if $\pi_i^{new} < \varepsilon$ for a threshold $\varepsilon > 0$, the inclusion is rejected. It is sensible to give J some "inertia": J for a new iteration should contain a

fraction of the top-scorers of the previous iteration. As long as d as small, we can use $L = \{1, \ldots, n\}$ and $J = \{1, \ldots, n\} \setminus I$. The cost for scoring all patterns in J is $O(|J|d^2)$.

4.4.2.3 The Extended Score Approximation

Extensions of the simple coupling-free selection score approximation have been motivated in Section 4.4.2.1 and are derived in Appendix C.4.3. While their use might lead to better selections of I, they require additional algorithmic details to be worked out and an exploration of this option in practice is left for future work. Nevertheless, we collect some remarks here.

The extended approximations require that a "neighbourhood set" H is given for each pattern i. Ideally, H is chosen such that $\boldsymbol{u}_H$ is correlated most strongly with u_i under the posterior, but computing this optimal neighbourhood structure is infeasible. A simple idea is to determine the structure a-priori, based on correlations under the GP prior. Given that the choice of the neighbourhood structure plays a subdominant role within PLV, any efficient clustering technique based on prior correlation as affinity measure will probably be sufficient. In order not to compromise our running time requirements, we are limited to hierarchical clustering schemes which represent subclusters by appropriate sufficient statistics. If the conditional inference algorithm is embedded into model selection it is sensible to re-select the structure now and then, since it depends on the kernel hyperparameters.

In the extended case, it is also more efficient to interleave parameter updates and score computations. For $j \in K$ to be scored let H be its neighbour set and $H' = H \cup \{j\}$. We compute $\boldsymbol{C}_{\cdot,H'}$, update the parameters of sites H' (j comes last) while keeping $\boldsymbol{C}_{\cdot,H'}$ up-to-date. Finally, the extended score is computed, passing $\boldsymbol{C}_{\cdot,H'}$. Again, the interleaving implies that we compare inclusion candidates based on different site parameter configurations, but keeping parameter updates and scoring separate would be much less efficient. K is chosen such that the overall number of required $\boldsymbol{C}$ column computations is acceptable. It should contain a fraction of top-scorers from the previous iteration.

The special case where site parameters are fixed from the beginning (Gaussian likelihood) is much simpler. $\boldsymbol{C}_{\cdot,H'}$ is not required for the score computation (see Appendix C.4.3.1), all we need is one back-substitution with Q. Even this can be saved if $\boldsymbol{C}$ is maintained explicitly (see Appendix C.4.1).

4.5 Model Selection

So far, we have discussed inference conditioned on free hyperparameters. A distinctive advantage of GP methods over many other kernel algorithms is that hyperparameters can be adapted by Bayesian marginal likelihood maximisation (see Sections 2.1.3 and A.6.2), allowing the use of kernels with a large number of free parameters. In this section, we show how model selection can be done for sparse GP approximations.

4.5.1 Model Selection for PLV

We follow the generic model selection scheme outlined in Section 2.1.3 (the hyperparameters are denoted by $\boldsymbol{\alpha}$). Note that the PLV scheme combines posterior approximations based on different optimality criteria. The EP scheme is based on moment matching, and an approximation to the marginal likelihood should be consistent with EP, i.e. fixed points of the EP scheme should correspond to saddle points of the approximation. For EP applied to full GP models, such a criterion can be constructed by generalising the Bethe free energy [215] as an analogous criterion for (loopy) belief propagation (see [123]). For the case of a Gaussian field with dense prior $P(\boldsymbol{u})$ and completely factorised likelihood $P(S|\boldsymbol{u})$ (as in Section 4.3), this criterion has been given in [41] and called ADATAP free energy. However, since PLV also uses KL-optimal projections which are based on e-projections, the ADATAP free energy does not apply in the same strong way to the sparse case. The usage of KL-optimal projections suggests the standard variational lower bound (A.18) as approximation, which is simpler to derive and handle than the ADATAP free energy.

From the derivation in Section 2.1.3 and the fact that $Q(\boldsymbol{u}_{\setminus I}|\boldsymbol{u}_I) = P(\boldsymbol{u}_{\setminus I}|\boldsymbol{u}_I)$

it is easy to see[16] that

$$\mathcal{G} = \mathrm{E}_Q\left[-\log P(\boldsymbol{y}|\boldsymbol{u})\right] + \mathrm{D}[Q(\boldsymbol{u}_I) \,\|\, P(\boldsymbol{u}_I)] \geq -\log \int P(\boldsymbol{y}|\boldsymbol{u}) P(\boldsymbol{u})\, d\boldsymbol{u}.$$

Here,

$$\mathrm{E}_Q\left[-\log P(\boldsymbol{y}|\boldsymbol{u})\right] = -\sum_{i=1}^{n} \mathrm{E}_Q\left[\log t_i(u_i)\right], \quad Q(u_i) = N(u_i|\mu_i, q_i + l_i^2)$$

by integrating out $\boldsymbol{u}_I$. In practice, we minimise the sum of $\mathcal{G}$ and negative log priors for the hyperparameters, the latter are suppressed here. Since $Q(\boldsymbol{u}_I) = N(\boldsymbol{L}\boldsymbol{Q}^{-T}\boldsymbol{\beta}, \boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T)$, $P(\boldsymbol{u}_I) = N(\boldsymbol{0}, \boldsymbol{L}\boldsymbol{L}^T)$, we can use Lemma A.13 to obtain

$$\mathrm{D}[Q(\boldsymbol{u}_I) \,\|\, P(\boldsymbol{u}_I)] = \frac{1}{2}\left(\log|\boldsymbol{M}| + \operatorname{tr}\boldsymbol{M}^{-1} - d + \left\|\boldsymbol{Q}^{-T}\boldsymbol{\beta}\right\|^2\right). \tag{4.17}$$

The gradient of $\mathcal{G}$ w.r.t. parameters of the covariance function and the noise model is derived in Appendix C.4.4.

The complete PLV scheme including model selection works as follows. In an outer loop, we minimise $\mathcal{G}$ w.r.t. the hyperparameters, using a gradient-based optimiser such as Quasi-Newton (see [55]). The optimiser requires evaluations of the criterion and its gradient for different purposes: to select new search directions, and to drive a line search along a fixed search direction. These computations are done by calling PLV conditional inference as a subroutine (see Section 4.4.2.2) whenever a new search direction is to be determined. During line searches, the configuration I, $\boldsymbol{b}$, $\boldsymbol{\Pi}$ is kept fixed. This optimisation scenario is discussed in more detail in Section 4.5.3.

If the true likelihood is Gaussian (say $N(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{\Pi}^{-1})$) we can use a simpler approximation:

$$-\log P(\boldsymbol{y}) \approx -\log \mathrm{E}_P\left[N(\boldsymbol{y}|\boldsymbol{\gamma}, \boldsymbol{\Pi}^{-1})\right], \quad P(\boldsymbol{\gamma}) = N(\boldsymbol{\gamma}|\boldsymbol{0}, \boldsymbol{V}^T\boldsymbol{V}). \tag{4.18}$$

This criterion which we use in our experiments in Section 4.8.2 is *not* equivalent to the variational $\mathcal{G}$ (details about the gradient computation for this criterion can be found in [165]). If the true likelihood is not Gaussian, this approximation does not apply because the likelihood approximations from EP need not correspond to a proper conditional distribution over the observables $\boldsymbol{y}$.

[16] Note that we minimise an upper bound here instead of maximising a lower bound, in order to remain compatible with standard formulations of optimisation algorithms.

4.5.2 Model Selection for IVM

In comparison to PLV, the IVM approximation to the true posterior is rather crude and model selection may suffer from this fact. On the other hand, IVM can be applied to significantly larger tasks for which principled approximate Bayesian model selection is rarely used.

We can in principle run the same scheme as for PLV above (Section 4.5.1): maximisation of a variational lower bound on the log marginal likelihood. However, doing so straightforwardly requires the complete $\boldsymbol{M}$ matrix (Equation C.5), so that randomised greedy selection (RGS) is not applicable. We can retain the randomisation advantage if we approximate the criterion even further. Let $L \subset \{1, \ldots, n\}$ with $I \subset L$. We use $\mathcal{G}$ in principle, but replace the true posterior by $\propto P(\boldsymbol{y}_L|\boldsymbol{u}_L)P(\boldsymbol{u})$, thus ignoring the data outside of L. If $L = \{1, \ldots, n\}$, this becomes the full criterion. Ideally, L is chosen such that the corresponding datapoints determine the shape of the true posterior most. Luckily, the greedy selection strategies we use for IVM are following exactly the same goal, so we can use information gathered during the selection of I in order to choose L. Recall from Section C.3.1.1 that RGS works by storing initial parts (stubs) of columns of $\boldsymbol{M}$. Whenever a pattern is to be scored, its stub has to be complete. Thus, if the stub cache is organised to (ideally) contain long stubs of patterns which render the best score values (or are likely to do so), it is sensible to include the points with largest available stubs into L. Of course, this also minimises the time required to complete $\boldsymbol{M}_{\cdot,L}$ which we need for the criterion computation. Again, an important advantage of the IVM scheme is that the running time of an iteration can be controlled by limiting the size of L. The details are developed in Appendix C.3.3.

4.5.3 Details about Optimisation

In this subsection, we comment on how criteria of the approximate log marginal likelihood type suggested above can be optimised using standard techniques.

The model selection problem as setup in this section boils down to minimise a function $f(\boldsymbol{\alpha})$ w.r.t. the vector of hyperparameters $\boldsymbol{\alpha}$ which is assumed to

be a real unconstrained vector.[17] However, if $\boldsymbol{\omega}$ collects all site parameters, $f(\boldsymbol{\alpha}) = \mathcal{F}\{g(\cdot; \boldsymbol{\alpha})\}$, where $\mathcal{F}$ operates on $g(\boldsymbol{\omega}; \boldsymbol{\alpha})$ by choosing the site parameters conditioned on a fixed $\boldsymbol{\alpha}$ using either PLV or IVM. Strictly speaking, $\mathcal{F}$ even has some random element, and because its computations involves the choice of I we cannot assume smooth behaviour w.r.t. $\boldsymbol{\alpha}$.

Even if the active set I is fixed, it seems very hard to compute the gradient $\nabla f(\boldsymbol{\alpha})$ because the mapping $\boldsymbol{\alpha} \mapsto \boldsymbol{\omega}$ is complicated. Thus, a straightforward gradient-based optimisation of f is not attractive. Our approach is to construct a sequence $(\boldsymbol{\omega}^{(t)}, \boldsymbol{\alpha}^{(t)})$ as

$$\boldsymbol{\alpha}^{(t+1)} = \operatorname*{argmin}_{\boldsymbol{\alpha}} g(\boldsymbol{\omega}^{(t)}, \boldsymbol{\alpha}), \quad \boldsymbol{\omega}^{(t+1)} = \mathcal{F}\{g(\cdot; \boldsymbol{\alpha}^{(t+1)})\},$$

where the minimisation is a line search along a search direction constructed from the gradient sequence $\boldsymbol{g}^{(t)} = \nabla_{\boldsymbol{\alpha}^{(t)}} g(\boldsymbol{\omega}^{(t)}, \boldsymbol{\alpha}^{(t)})$, using a standard method such as conjugate gradients or Quasi-Newton (see [55]). This is motivated by $\boldsymbol{g}^{(t)}$ being an approximation to $\nabla f(\boldsymbol{\alpha}^{(t)})$ (see Section 2.1.3).

This approach is closely related to variational EM, another variant of lower bound maximisation (see Section A.6.4). However, the latter case has nicer properties, due to the fact that $\mathcal{F}\{g(\cdot; \boldsymbol{\alpha})\} = \operatorname{argmin}_{\boldsymbol{\omega}} g(\boldsymbol{\omega}; \boldsymbol{\alpha})$. First, the gradients $\boldsymbol{g}^{(t)}$ are exact in this case, furthermore every step is guaranteed to decrease the objective f, given that the search directions have positive inner product with the gradients there. Neither of these properties is true in general for our approach here. If

$$f^{(t)} = f(\boldsymbol{\alpha}^{(t)}) = g(\boldsymbol{\omega}^{(t)}; \boldsymbol{\alpha}^{(t)}),$$

we might encounter $f^{(t+1)} > f^{(t)}$. In practice, oscillatory behaviour is observed during later stages and convergence to high relative accuracy cannot be expected. We note that both variational EM and our approach here do not minimise f during line searches, but $g(\boldsymbol{\omega}^{(t)}, \cdot)$.[18] Thus, methods for constructing the sequence of search directions which rely on fairly accurate line minimisations, such as

[17] Positivity constraints can be enforced by using log transformations.

[18] Can we make line searches more efficient by updating $\boldsymbol{\omega}$ in between? This might lead to problems in practice, because the dependence of $\boldsymbol{\omega}$ on $\boldsymbol{\alpha}$ is ignored during gradient computation, so during the line search finite differences for small steps will not match corresponding derivatives (projections of the gradient).

conjugate gradients, can only be used in a modified form. For example, more frequent restarts (with the gradient as search direction) are recommended. Quasi-Newton methods are known to rely less on accurate line searches.

However, even if our approach does not share the monotone convergence properties (to a local minimum) with variational EM, the optimisation problem is neither a very hard one. Search directions based on the gradient approximations $\boldsymbol{g}^{(t)}$ seem to work efficiently in practice. Moreover, our optimisation code incorporates a recover logic which restarts whenever a line search fails or a proposed direction is not a descent one. A critical problem in practice is the choice of a suitable stopping criterion. The relative improvement in criterion value for successive iterations is not a suitable measure in this context due to oscillatory behaviour. If we choose two lags $T_1 \leq T_2 > 0$ and define

$$M^{(T,t)} = \min\{f^{(t-T+1)}, \ldots, f^{(t)}\},$$

we can monitor

$$\Delta^{(t)} = \frac{M^{(T_1,t-T_2)} - M^{(T_1,t)}}{|M^{(T_1,t-T_2)}|}.$$

For a non-increasing sequence $f^{(t)}$, $\Delta^{(t)}$ is non-negative for any T_1, T_2. In our context, "convergence" can be assessed based on the size and sign of $\Delta^{(t)}$ for not too small lags, possibly in combination with checking for small $\|\boldsymbol{g}^{(t)}\|$. Any such stopping criterion in practice means trading off unnecessarily long running time[19] against the danger of early spurious convergence (see Section 4.8.4 for examples). An empirical study to find an efficient stopping criterion for the model selection problems of this section is subject to future work.

To conclude, the fact that standard optimisation packages may not be directly applicable to the model selection problem, or may result in unnecessary inefficiencies is certainly a negative aspect of our strategy. On the other hand, the modifications required are relatively minor: essentially, it has to be possible to use different criteria during line searches than for computing new search directions and assessing convergence, and robust behaviour in the presence of minor

[19]Given that the optimisation behaves stable close to an optimum. This was the case in all our experiments.

oscillations is needed. Furthermore, nowadays off-the-shelf programming packages are rarely used to solve the problems of very particular structure posed by kernel methods such as SVM, because specialised code can run much faster.

4.6 Related Work

Sparse approximations to kernel methods have received a lot of recent attention in the machine learning community. In this section, we review schemes which are related to the work presented in this chapter.

As already mentioned above, our methods are most closely related to the work of Csató and Opper. The PLV scheme differs from the algorithm in [41] in that we use greedy forward selection to decide about inclusions, while the latter iterates over the data in an on-line fashion (although the algorithm is not an on-line scheme), deciding inclusions or removals based on approximation error criteria. Model selection based on minimising the ADATAP free energy is not reported in [38], but has been done since then. Greedy forward selection can be more efficient than a random traversal approach if an efficient selection criterion is able to pick out informative points early. Also, schemes which repeatedly remove and include points may be prone to numerical instability, since matrix update formulae like Woodbury (Lemma A.2) are notoriously ill-behaved especially if the rank-one addition is negative definite. The IVM is related to the sparse on-line scheme of [40, 39], but is simpler and slightly more efficient. Csató and Opper [40] use a seemingly more direct representation than ours (see Section C.3.1). However, both representations are equivalent in terms of running time, storage requirements and numerical stability, while ours has the advantage of allowing for randomised greedy selections which can speed up the algorithm significantly (see Section C.3.1.1). Because a factorised likelihood approximation is used, the greedy selection criteria of Section 4.2.2 can be computed in $O(1)$ (given the representation) for the IVM. In contrast, the on-line scheme uses a likelihood which couples the sites and therefore may lead to more accurate approximations. The IVM is not an on-line scheme. Csató [38] applies the sparse GP approximation

technique to a wide range of applications, including binary classification, density estimation and wind-field prediction.

The special case of PLV for regression with Gaussian noise [165] is related to sparse greedy GP regression [183] and Hybrid Adaptive Splines [107]. All these methods use greedy forward selection to choose patterns (or their corresponding "basis functions" $K(\cdot, \boldsymbol{x}_i)$) for I. Given the model and the active set I, the optimal predictor can be derived straightforwardly, thus the important difference for sparse greedy GP regression methods lies in the selection criterion. Both Smola and Bartlett [183] and Luo and Wahba [107] use what we called expensive criteria in Section 4.2.2: the scoring of a point scales in the same way as its inclusion, namely $O(n\,d)$. In our opinion, this is prohibitively expensive, since in each round only a small number of candidates can be scored, and even then the total training time is strongly dominated by the score computations. In contrast, our approximate information gain score (see Sections 4.4.2.1, C.4.2, C.4.3) can be evaluated in $O(h\,d)$,[20] where h can be chosen anywhere between 1 and n, or even in $O(1)$ if all couplings for the new point are ignored. Furthermore, neither Smola and Bartlett nor Luo and Wahba consider generalisations to arbitrary GP models, and both recommend cross-validation schemes to select free hyperparameters which can be prohibitive if there are many. The Smola/Bartlett scheme can easily be described in our notation by transforming to dual variables[21] $\boldsymbol{\alpha} = \boldsymbol{K}^{-1}\boldsymbol{u}$ (recall our notation from Section 4.4.2). Their scheme is a sparse MAP approximation, in that they minimise

$$\begin{aligned}\pi(\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{\alpha}^T(\sigma^2\boldsymbol{K} + \boldsymbol{K}^T\boldsymbol{K})\boldsymbol{\alpha} - \boldsymbol{y}^T\boldsymbol{K}\boldsymbol{\alpha} \\ &= -(\boldsymbol{V}\boldsymbol{y})^T\boldsymbol{L}^{-1}\boldsymbol{u}_I + \frac{\sigma^2}{2}\boldsymbol{u}_I^T(\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T)^{-1}\boldsymbol{u}_I,\end{aligned}$$

where $\boldsymbol{\alpha}$ is sparse in the sense that $\boldsymbol{\alpha}_{\setminus I} = \boldsymbol{0}$. $\pi(\boldsymbol{\alpha})$ is proportional to the negative log posterior $-\log P(\boldsymbol{u}_I|\boldsymbol{y})$ and is minimised by the posterior mean $\sigma^{-2}\boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{V}\boldsymbol{y}$, the minimum value is

$$\pi_I = \min_{\boldsymbol{\alpha}} \pi(\boldsymbol{\alpha}) = -\frac{\sigma^2}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}.$$

[20] Note that in the regression case, we do not have to update site parameters.

[21] The $\boldsymbol{\alpha}$ here has nothing to do with the hyperparameter vector $\boldsymbol{\alpha}$ of Section 4.5.

Their selection heuristic for a point $i \notin I$ is

$$\Delta_i^{\mathrm{SB}} = \pi_I - \pi_{I'} = \frac{\sigma^2}{2}\beta_{d+1}^2, \quad I' = I \cup \{i\},$$

i.e. the decrease that can be obtained in $\pi(\boldsymbol{\alpha})$ by allowing component i in $\boldsymbol{\alpha}$ to become non-zero. As shown in Section C.4.1, the computation of β_{d+1} requires two $n \cdot d$ computations. Smola and Bartlett consider a randomised scheme in which Δ_i^{SB} is evaluated only over a small random subset of size k (the authors recommend $k = 59$), leading to a training complexity of $O(n\,k\,d^2)$ which is k times more expensive than PLV (for regression). Also, since the random subsets fluctuate freely over $\{1, \ldots, n\}$, one typically ends up evaluating a large fraction of the full kernel matrix $\boldsymbol{K}$ which is problematic if kernel evaluations are expensive. In contrast, PLV requires $\boldsymbol{K}_{\cdot,I}$ only (I the final active set).

A number of authors suggest using low-rank matrix approximations to end up with sparse versions of conventional kernel algorithms [181, 208, 211, 54, 129, 134, 200]. As mentioned in Section 2.1.3, many GP approximations are in terms of a matrix $(\boldsymbol{K}^{-1} + \boldsymbol{D})^{-1}$, where $\boldsymbol{D}$ is diagonal and positive, the same holds for some SVM solvers. Given an active set $I \subset \{1, \ldots, n\}$, both Smola and Schölkopf [181] and Williams and Seeger [208, 211] suggest replacing $\boldsymbol{K}$ by the low-rank approximation $\boldsymbol{K}_{\cdot,I}\boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,\cdot} = \boldsymbol{V}^T\boldsymbol{V}$, giving different theoretical justifications for this choice (see also Section 4.7.1). Then,

$$\left(\boldsymbol{K}^{-1} + \boldsymbol{D}\right)^{-1} \approx \left(\boldsymbol{V}^T\boldsymbol{V} + \boldsymbol{D}\right)^{-1} = \boldsymbol{D}^{-1} - \boldsymbol{D}^{-1}\boldsymbol{V}^T\left(\boldsymbol{I} + \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{V}^T\right)^{-1}\boldsymbol{V}\boldsymbol{D}^{-1},$$

requiring a d-by-d Cholesky decomposition only. Plugging this approximation into a given algorithm renders a scaling of $O(n\,d^2)$. While Williams and Seeger suggested a random choice of I (see also [61, 129]), Smola and Schölkopf use greedy forward selection with the Frobenius norm as criterion (this is also an expensive criterion in the sense of Section 4.2.2). A different approach is used in [54], where a Cholesky decomposition of the target matrix (e.g., $\boldsymbol{K}$) with row pivoting[22] is stopped early to obtain a low-rank approximation. The advantage of this method is that finding the largest pivot is very fast, it corresponds to

[22] Row pivoting is the essential feature here. Note that pivoting is not required for a complete Cholesky decomposition, since this would not improve numerical stability.

greedy forward selection with the criterion $\mathrm{tr}(\boldsymbol{K} - \tilde{\boldsymbol{K}})$ if $\tilde{\boldsymbol{K}}$ denotes the low-rank approximation. Although this trace criterion is always non-negative, it is not very clear what it means in terms of approximation quality (the authors present some bounds which are however extremely loose). In contrast, the Frobenius norm $\mathrm{tr}(\boldsymbol{K} - \tilde{\boldsymbol{K}})^2$ used in [181] comes with strong approximation guarantees, but does not lead to an efficient method. The incomplete Cholesky approximation has been used in [8], and variants are routinely used in interior-points code for linear programming [212]. As already noted in [107], all these methods share a common problem: their choice of I does depend only on the input datapoints, the target values $\boldsymbol{y}$ are completely ignored (in fact, the low-rank decompositions can be computed without knowing $\boldsymbol{y}$).

A powerful technique for sparse Bayesian GP learning is Tipping's *relevance vector machine (RVM)* [192, 52]. The idea is to consider a kernel expansion $u(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{k}(\boldsymbol{x})$ and sparsify the weight vector $\boldsymbol{w} \in \mathbb{R}^n$ using *automatic relevance determination (ARD)* ([131]; see also Section 2.1.8). By placing a prior $P(\boldsymbol{w}|\boldsymbol{\alpha}) = N(\boldsymbol{w}|\boldsymbol{0}, \mathrm{diag}\, \boldsymbol{\alpha}^{-1})$ on $\boldsymbol{w}$ and an improper uniform prior on the components of $\log \boldsymbol{\alpha}$, then learning $\boldsymbol{\alpha}$ by maximising the hyperposterior $P(\boldsymbol{\alpha}|S)$ (this is a variant of marginal likelihood maximisation, see Section A.6.2) one finds that on real-world data many of $\boldsymbol{\alpha}$'s components tend to ∞, thus fixing the corresponding components of $\boldsymbol{w}$ (in the approximate posterior) to 0. The original RVM cannot be considered a sparse approximation technique in the sense of Section 4.1, because it requires $O(n^3)$ training time (it is actually more expensive to train than most methods mentioned in Section 2.1.3), but later refinements have shown how to do RVM training incrementally [52]. The authors recommend a joint maximisation of the marginal likelihood (or the posterior) w.r.t. the sparsity hyperparameters $\boldsymbol{\alpha}$ and additional kernel parameters, although they do not provide experimental results in [52]. The sampling mechanism used to motivate the RVM does not correspond to a proper GP model (or in fact to any process model). It implies f.d.d.'s $N(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}(\mathrm{diag}\, \boldsymbol{\alpha})^{-1}\boldsymbol{K})$ for $u(\boldsymbol{x})$ conditioned on $\alpha(\boldsymbol{x})$ which are not consistent in the sense of Kolmogorov's criterion (see Section 2.1.1). Our view on RVM is that once $\boldsymbol{w}$ is integrated out, the search for $\boldsymbol{\alpha}$ coefficients

is a particular way to construct a sparse Gaussian posterior approximation (i.e. $\boldsymbol{\alpha}$ parameterises the posterior approximation just as for example $\boldsymbol{D}$ does in Section 2.1.3). When viewed like this, RVM falls into the same class as the schemes described in this chapter, although the Bayesian ARD motivation for selecting $\boldsymbol{\alpha}$ is weakened then. It is not clear *a priori* how the selection of I via greedy maximisation of the marginal likelihood compares with the information-based criteria of Section 4.2.2, an experimental comparison may be required to gain clarity here. A version of RVM for binary classification is suggested in [192], using a Laplace approximation (see Section 2.1.3). An older scheme related to RVM is the *subset of regressors (SR)* model [200], where given I we consider the sparse kernel expansion $u(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{k}_I(\boldsymbol{x}),\ P(\boldsymbol{w}) = N(\boldsymbol{0}, \boldsymbol{K}_I^{-1})$. From a Bayesian viewpoint, this model is unusual in that the prior on $\boldsymbol{w}$ depends on the input points and also on the selection of I, i.e. a particular detail of inference approximation. In fact, the predictive variance $\mathrm{Var}[u(\boldsymbol{x})|S]$ tends to 0 far from the training inputs for many isotropic standard kernels, in stark contrast to proper GP methods for which the variance becomes maximal then. To be fair, the SR model was probably never intended to be a Bayesian one. The finite-dimensional Bayesian formulation for the generalised spline smoothing method of Green and Silverman discussed in Section 2.1.5 suffers from the same problem.

The *Bayesian committee machine (BCM)* of [194] is a different approach, in that the training sample S is split into a number of blocks of size d (say), then the n/d predictors based on the subsamples are combined in a somewhat Bayesian way. The argument is that conditioned on some query points which may be the test set or a part of it, the subsamples become approximately conditionally independent. Under this assumption, the committee renders an approximation to the joint posterior over the query points. The advantage of the BCM over other sparse methods discussed so far may be that all data is actually used for prediction. It does not render an approximation to the predictive distribution which is a Gaussian process. One could use a single test point as query set, in which case however the conditional independence assumption seems too strong. Otherwise, prediction requires bundling of test points into query sets which can

be inconvenient. Furthermore, in predictions we are usually interested only in marginal distributions at test points, while the BCM tries to approximate the whole joint distribution over the query points. The BCM scales cubically in d and either in the size of the query set or in n/d. The *infinite mixture of Gaussian process experts* [148] is another method which uses a committee of small GP predictors. It is an MCMC method, based on sampling from a Dirichlet process (see also [147]), thus in theory leads to correct Bayesian inference in the limit. However, on high-dimensional real-world data we would not expect the Markov chain on this model to mix in reasonable time, for example the incentive of an indicator variable to switch experts should be very low. The authors present experimental results on low-dimensional data only.

Tong and Koller [193] consider active learning strategies for support vector machines (SVMs). It has been noted (see Section 4.2.2 or [38], Sect. 5.2) that sparse GP approximations like IVM or PLV could be applied to active learning rather easily, due to the simple forms of the criteria of Section 4.2.2. In contrast to this, Tong and Koller's version space[23] oriented approach seems fairly elaborate. The MaxMin approach seems interesting, but requires retraining the SVM twice (including one more point) in order to evaluate the criterion.

Sparse approximations of the solution to smoothing spline problems (see Section 2.1.5) is discussed in [134], see also [200], Chap. 7. The optimisation of the penalised risk functional is restricted to functions in the span of a finite number of basis functions which are chosen in a data-dependent way. Once a given set of "representative" input points[24] $\boldsymbol{s}_i$ is chosen (of size much smaller than n), the basis functions are constructed as linear combinations of the $K(\cdot, \boldsymbol{s}_i)$, where K is the kernel for the RKHS. This coincides in principle with the methods discussed in this chapter as long as the $\boldsymbol{s}_i$ are chosen from the training set (the BCM includes test input points as well), which is arguably a good strategy if little is known about the input point distribution. For special spline kernels K

[23] In this context, the version space is the part of the hypersphere corresponding to the normal vectors of all hyperplanes which classify all data points correctly.

[24] The $\boldsymbol{s}_i \in \mathcal{X}$ need not come from the training set and are therefore denoted by different symbols.

and low-dimensional input spaces, the basis function can be chosen to have compact support (so-called B-splines), which can lead to advantages in the numerical optimisation if the $\boldsymbol{s}_i$ are spread out.

The approximation of a GP by a linear model with less than n components has been studied in [135, 216]. The former method requires $O(n^3)$ computations, while Zhu et. al. [216] propose an *a priori* optimal approximation based on the Mercer eigen-expansion of the kernel (see Section 2.1.4) and the input distribution (which is assumed to be known).

Finally, as mentioned in Section 2.1.6 GP priors have been used to drive methods from two quite different paradigms, the probabilistic Bayesian one and the large margin discriminative one. While in this chapter we have focussed on sparse Bayesian GP learning, the same ideas can be applied to sparse variants of large margin algorithms such as SVM. For example, the framework developed in this chapter could be applied straightforwardly to the MRED variant of SVM (see Section 2.1.6). Indeed, since we do not have to worry about estimating predictive probabilities, the resulting method would even be somewhat simpler to implement (although we may argue that by estimating these probabilities we can use more effective scores to select new inclusion candidates). For the standard SVM, such incremental learning has been proposed in [27]. The idea is to restore the KKT conditions (see Section 2.1.6 and [161]) after including a new point along with its dual variable using "adiabatic" increments in order to retain these conditions on all other dual variables while working towards satisfying them on the new pattern as well. The method seems to rely crucially on the dual to be a quadratic function so that its gradient is linear in the dual variables which allows to compute the exact effects of non-infinitesimal variations.[25] Since the authors aim to learn all support vectors eventually, they do not report using forward selection strategies, but it would be straightforward to use (normalised) margin-based ones (Tong and Koller [193] suggest criteria for SVM active learning which could be adapted). However, the information-theoretic criteria we have used here are not applicable,

[25]Note that for the same reason, the cross-validation score (or generalised CV score) can be computed without re-training n times for the generalised spline smoothing procedure of Section 2.1.5, while this is not possible for general penalised likelihood methods.

due to the failure of the SVM to estimate predictive variances.

4.7 Addenda

In this section, we collect some additional material which complements the main parts of this chapter.

4.7.1 Nyström Approximations

In this section, we briefly review the basic ideas of [211, 208] and some related work, details can be found in these references.

Recall from Section 2.1.4 that for every positive definite kernel K, there exists a uniquely determined Hilbert space (HS) $\mathcal{H}_K$ admitting K as reproducing kernel (RK): the reproducing kernel Hilbert space (RKHS) for K. Also recall the eigen-expansion of K (2.19) in terms of orthonormal eigenfunctions in $\mathcal{L}_2(\mathcal{X})$. We are free in choosing the weighting measure for $\mathcal{L}_2$. For a positive density function $p(\boldsymbol{x})$ on $\mathcal{X}$ consider the modified inner product

$$(f, g)_p = \int p(\boldsymbol{x}) f(\boldsymbol{x}) g(\boldsymbol{x})\, d\boldsymbol{x} = \mathrm{E}_{\boldsymbol{x}}[f(\boldsymbol{x})g(\boldsymbol{x})].$$

An eigen-expansion of K exists if $\operatorname{tr} K^2 = \int\int p(\boldsymbol{x})p(\boldsymbol{y})K^2(\boldsymbol{x}, \boldsymbol{y})\, d\boldsymbol{x}d\boldsymbol{y} < \infty$. The Hilbert space $\mathcal{H}$ with this inner product is isometrically isomorphic to $\mathcal{L}_2(\mathcal{X})$ (with Lebesgue measure) under the mapping $f \mapsto p^{1/2} f$. It is easy to see that if K is the RK on $\mathcal{H}$, then $\tilde{K}(\boldsymbol{x}, \boldsymbol{y}) = p(\boldsymbol{x})^{1/2} K(\boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{y})^{1/2}$ is the RK on $\mathcal{L}_2(\mathcal{X})$.

In the usual applications of Gaussian processes or spline smoothing methods, it is not necessary to know the eigen-representation of the kernel K w.r.t. any inner product, but if the original underlying variational problem is to be *approximated* by concentrating efforts on a lower-dimensional span of basis functions, the eigenfunctions of K become important. In order to be able to estimate the eigenfunctions from data without certain knowledge of the true data distribution $p(\boldsymbol{x})$, we use $\mathcal{H}$ as above:

$$\lambda_i \phi_i(\boldsymbol{y}) = \mathrm{E}_{\boldsymbol{x}}\left[K(\boldsymbol{x}, \boldsymbol{y})\phi_i(\boldsymbol{x})\right] \approx \frac{1}{n}\sum_{k=1}^{n} K(\boldsymbol{x}_k, \boldsymbol{y})\phi_i(\boldsymbol{x}_k) \tag{4.19}$$

for large n. A standard approach is to plug in $\boldsymbol{y} = \boldsymbol{x}_j$ to obtain the n-dimensional eigenproblem $\boldsymbol{K}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}$, where $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{\Lambda} = \text{diag}(\hat{\lambda}_j)_j$, $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq 0$. We can use this to approximate $\phi_i(\boldsymbol{x}_j) \approx \sqrt{n}\boldsymbol{U}_{i,j}, \quad \lambda_i \approx n^{-1}\hat{\lambda}_i$, where we have used $n^{-1}\sum_k \phi_i(\boldsymbol{x}_k)\phi_j(\boldsymbol{x}_k) \approx \mathrm{E}_{\boldsymbol{x}}[\phi_i(\boldsymbol{x})\phi_j(\boldsymbol{x})] = \delta_{i,j}$. We would expect this approximation to be most accurate for small i and deteriorate for i close to n. Plugging this back into (4.19), we obtain the *Nyström approximation*

$$\phi_i(\boldsymbol{y}) \approx \hat{\phi}_i(\boldsymbol{y}) = \frac{\sqrt{n}}{\hat{\lambda}_i}\sum_{k=1}^{n} K(\boldsymbol{y}, \boldsymbol{x}_k)\boldsymbol{U}_{k,i} = \frac{\sqrt{n}}{\hat{\lambda}_i}\boldsymbol{U}_{\cdot,i}^T\boldsymbol{k}(\boldsymbol{y}),$$

where $\boldsymbol{k}(\boldsymbol{y}) = (K(\boldsymbol{x}_j, \boldsymbol{y}))_j$. It is known that for any fixed k, $n^{-1}\hat{\lambda}_k \to \lambda_k$ almost surely as $n \to \infty$, for details see [9].

Why would a knowledge of the dominating eigenfunctions be useful for approximation? Recall the Karhunen-Loeve expansion (2.20) for a zero-mean Gaussian process $u(\boldsymbol{x})$ with covariance K. We note that most variability in this prior ensemble is concentrated on the space of eigenfunctions of the leading eigenvalues, given that the spectrum decays reasonably quickly. In [208] it is argued that at least in the GP regression setting, the coefficient u_ν of eigenfunction ϕ_ν in the (predictive) posterior process is "damped" to be close to zero with high probability if $n\lambda_\nu$ is smaller than the noise variance, at least if the data is indeed sampled from this model. Thus, the eigenfunctions associated with the leading eigenvalues will typically dominate predictive estimates, therefore are prime candidate for building a basis for a reduced approximation.[26] A more thorough and explicit analysis is given in [216]. Both the spectrum and the eigenfunctions depend crucially on the distribution $p(\boldsymbol{x})$ of the data, and leading entries can often be estimated reliably from large samples as mentioned above. In [208], the dependence of spectrum and eigenfunctions on $p(\boldsymbol{x})$ is illustrated in 1-d for the case of the Gaussian kernel (2.29) and $p(\boldsymbol{x})$ a mixture of well-separated Gaussians.

The Nyström approximation suggests approximating the GP with covariance $K(\boldsymbol{x}, \boldsymbol{y})$ by a linear model with basis functions $\hat{\phi}_i(\boldsymbol{y})$, $i = 1, \ldots, p$, $p \ll n$. This

[26] This is essentially the same argument as used to motivate PCA.

is equivalent to approximating the kernel function by

$$\sum_{i=1}^{p} \hat{\lambda}_i \hat{\phi}_i(\boldsymbol{x}) \hat{\phi}(\boldsymbol{x}') = \boldsymbol{k}(\boldsymbol{x})^T \boldsymbol{U}_{\cdot,1\ldots p} \boldsymbol{\Lambda}_{1\ldots p}^{-1} \boldsymbol{U}_{\cdot,1\ldots p}^T \boldsymbol{k}(\boldsymbol{x}'),$$

and it means that the kernel matrix over the training data is approximated by $\boldsymbol{U}_{\cdot,1\ldots p} \boldsymbol{\Lambda}_{1\ldots p} \boldsymbol{U}_{\cdot,1\ldots p}^T$, which is closest to $\boldsymbol{K}$ in Frobenius norm[27] among all rank-p matrices. This approximation is somewhat related to the method suggested in [63], where a Lanczos algorithm is used to find $\boldsymbol{U}_{\cdot,1\ldots p}$ approximately. The problem is that finding $\boldsymbol{U}_{\cdot,1\ldots p}$ is hard if p is not extremely small, and in any case requires at least $O(n^2 p)$ and computation and storage of the complete $\boldsymbol{K}$. In [211], it was suggested to use the Nyström approximation once more to approximate $\boldsymbol{U}_{\cdot,1\ldots p}$ based on a subset $\{\boldsymbol{x}_i \,|\, i \in I\}$. After an eigen-decomposition of $\boldsymbol{K}_I \in \mathbb{R}^{d,d}$, the remaining values $\boldsymbol{U}_{\setminus I,1\ldots p}$ are filled in using the approximation $\hat{\phi}_i(\boldsymbol{x}_k), i \in \{1,\ldots,n\} \setminus I$. If $p = d$, this is equivalent to replacing the covariance matrix $\boldsymbol{K}$ by $\boldsymbol{K}_{\cdot,I} \boldsymbol{K}_I^{-1} \boldsymbol{K}_{I,\cdot}$ which can be motivated from many other angles as well (for example, see Section 4.4.2). The index set I was picked at random in [211], in contrast to our emphasis on greedy forward selection strategies in this chapter.

Low-rank approximations of kernel matrices are used in [8] to provide sample-based efficient approximations to the mutual information. Their accuracy hinges on a sufficiently fast decay of the kernel matrix eigenspectrum, and the authors review some work which tries to quantify the decay. For stationary kernels, the rate of decay of the covariance operator spectrum depends on the tail behaviour of the spectral density of K (see Section 2.1.1) and of the input density. Both [208] and [8] contain empirical studies comparing the theoretical decay rates of the operator eigenvalues with the Nyström approximations computed from actual kernel matrices and both find that the approximations tend to underestimate the process eigenvalues. The theoretical rates for stationary kernels in higher dimensions predict a much slower decay than in 1-d. However, since real-world data in high dimensions often lies close to a low-dimensional manifold, this does not necessarily imply a slow decay of kernel matrix spectra for such data.

Shawe-Taylor and Williams [175] provide some concentration results which

[27] The Frobenius norm is $\|\boldsymbol{A}\|_2 = (\operatorname{tr} \boldsymbol{A}^T \boldsymbol{A})^{1/2}$.

lead to gap bounds between matrix and process eigenvalues. The idea is to relate sums of leading eigenvalues of covariance matrix and covariance operator, using their respective Courant-Fisher characterisation. Recall from Section 2.1.4 that if an eigen-decomposition of K in $\mathcal{H}$ with inner product $(f, g)_p$ is given, the RKHS admitting K embedded in $\mathcal{H}$ can be characterised via the Fourier coefficient vectors of $f \in \mathcal{H}$, namely $f_i = (f, \phi_i)_p = \mathrm{E}[f(\boldsymbol{x})\phi_i(\boldsymbol{x})]$ and $\boldsymbol{f} = (f_i)_i$. By the Courant-Fisher min-max theorems, we can relate $n^{-1}\sum_{i=1}^k \hat{\lambda}_i$ for leading eigenvalues of $\boldsymbol{K}$ with $\sum_{i=1}^k \lambda_i$ for leading process eigenvalues. The latter is the maximum over all k-dimensional subspaces of the feature space of $\mathrm{E}[\|\boldsymbol{P}(\psi(\boldsymbol{x}))\|^2]$, where $\boldsymbol{P}$ denotes the projection onto the subspace, and $\|\boldsymbol{f}\|^2 = \boldsymbol{f}^T\boldsymbol{f}$ in feature space. The maximising subspace is spanned by the leading k eigenfunctions $\phi_i,\ i = 1, \ldots, k$. To relate this to the matrix eigenvalues, we observe that if $\boldsymbol{X} = (\psi(\boldsymbol{x}_1) \ldots \psi(\boldsymbol{x}_n))$ is the data matrix in feature space, then $\boldsymbol{K} = \boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{X}\boldsymbol{X}^T$ have the same leading eigenvalues, and for the latter we can use the Courant-Fisher theorem in feature space just as in the population case, simply replacing expectations over $p(\boldsymbol{x})$ by such over the empirical distribution $\hat{p}(\boldsymbol{x}) = n^{-1}\sum_j \delta_{\boldsymbol{x}_j}(\boldsymbol{x})$. The maximising subspace is different from the one in the population case and can be constructed from the matrix eigen-decomposition. The min-max characterisation now allows to infer various inequalities, and for the fixed subspace based on the process eigenfunctions, we can use concentration results to bound the difference between empirical and population expectation.

4.7.2 The Importance of Estimating Predictive Variances

In this section, we focus on the binary classification problem. Recall from Section 2.1.3 that the class of approximations to Bayesian inference for GP models we are interested in here aims to approximate the posterior process by a Gaussian one. This second-order approach naturally leads to predictions of the variance of this process (2.14) quantifying local uncertainty in the posterior mean prediction[28] $\mu(\boldsymbol{x}_*)$. On the other hand, valid estimates for predictive uncertainty are

[28] If there is a bias parameter b, we assume it has already been added to $\mu(\boldsymbol{x}_*)$ in the sense that our point predictions are $\operatorname{sgn}\mu(\boldsymbol{x}_*)$.

not obtained from large margin discriminative methods (Section 2.1.6) such as SVM. The SVM discriminant output (say $s(\boldsymbol{x}_*)$) could be related to $\mu(\boldsymbol{x}_*)$ since decisions in SVM and IVM are based on $\operatorname{sgn} s(\boldsymbol{x}_*)$ and $\operatorname{sgn} \mu(\boldsymbol{x}_*)$ respectively, but since SVM does not constitute a proper probabilistic model (see Section 2.1.6) there is no reason why $s(\boldsymbol{x}_*)$ should be a useful posterior mean prediction. In fact, the SVM loss has some peculiar characteristics which are essential for a large margin discriminative method (see Section 2.1.6): it is not differentiable at $+1$, has zero curvature elsewhere and is flat zero for $y\,s \geq 1$. Even if it corresponded to a negative log likelihood, we would not consider it a sensible model for classification noise. The ability to produce valid uncertainty estimates comes with additional cost: both training and prediction time are somewhat longer for Bayesian GP methods than for comparable SVM techniques (at least for binary classification).

Platt [140] proposed to obtain predictive probabilities by ad-hoc post-processing. Namely,

$$\Pr\{y = +1|\boldsymbol{x}, Q\} = \sigma(a\, s(\boldsymbol{x}) + b)$$

where $\sigma(x)$ is the logistic function (see Equation 2.8). The scalars a, b are fitted post-hoc by regularised maximum likelihood using some "unbiased version" of the training set. Clearly, these outputs are in $(0, 1)$ and can be sold as probabilities, but the overall validity of the method remains unclear. First, if the SVM is assumed to perform well it is not sensible to choose $b \neq 0$, otherwise the obvious decision rule based on the predictive probabilities would be different from the SVM decision (and usually perform worse). Next, if $a > 0$ (we suppose this is intended) the predictive probability is strictly increasing in $s(\boldsymbol{x})$. If we define

$$m(\boldsymbol{x}) = |\Pr\{y = +1|\boldsymbol{x}, Q\} - 1/2| \tag{4.20}$$

then our uncertainty in a point prediction decreases with increasing $m(\boldsymbol{x})$. But a ranking of a test set w.r.t. this uncertainty is exactly the same for $m(\boldsymbol{x})$ and for $|s(\boldsymbol{x})|$ (given that $b = 0$). Thus, for ranking as one of the most important applications of predictive probabilities (see below) Platt's estimate does not provide anything new. In fact, all "order-related" problems of $|s(\boldsymbol{x})|$ are fully transferred

to $m(\boldsymbol{x})$. Apart from these obvious problems, a number of additional heuristics are required to make it work in practice. Tipping [192] pointed out similar weaknesses.

Point predictions without uncertainty estimates would be considered an incomplete analysis by many statisticians, even for classification. On the other hand, predictive probabilities are important for all but the simplest decision problems. For example, they are required if decisions result in unequal losses (or utilities). A very important argument for proper probabilistic models is that they can be combined and integrated in a simple and consistent way in larger systems, simply by the consistency of elementary probability laws.[29] Here, we concentrate on a simple decision problem in which we have a reject option for an α fraction of a set of test points. We use the experimental results described in Section 4.8.1 for the hardest binary task $c = 9$ against the others. Figure 4.4 shows an error-reject curve for IVM and SVM (obtained by averaging curves over 10 runs). From about $\alpha = 1/20$, the error rate after rejection is significantly lower for IVM. Also (not shown here), the variability in the curves over the 10 runs is larger for SVM. Recall that the test points are ranked using $|s(\boldsymbol{x}_*)|$ for SVM and $m(\boldsymbol{x}_*)$ (4.20) for IVM. For the probit noise model (2.9), if we define

$$r(\boldsymbol{x}_*) = \frac{\mu(\boldsymbol{x}_*)}{\sqrt{1 + \sigma^2(\boldsymbol{x}_*)}}$$

then

$$m(\boldsymbol{x}_*) = \left|\Phi(r(\boldsymbol{x}_*)) - \Phi(0)\right|,$$

and since $\Phi(x) - 1/2$ is odd the IVM ranking can be based on $|r(\boldsymbol{x}_*)|$ as well. Note the explicit role of the predictive variance $\sigma^2(\boldsymbol{x}_*)$ here: the larger $\sigma^2(\boldsymbol{x}_*)$ (thus our uncertainty in $\mu(\boldsymbol{x}_*)$ as prediction of $\mathrm{E}[u(\boldsymbol{x}_*)]$), the more $r(\boldsymbol{x}_*)$ is shrunk towards 0. If we believe that SVM provides a good prediction of the posterior mean, it should lead to a similar ranking than one based on $|\mu(\boldsymbol{x}_*)|$ for the IVM. Figure 4.1 shows error-reject curves for all three ranking criteria.

We see that SVM still performs worst, but is closer to the (also invalid) ranking via $|\mu(\boldsymbol{x}_*)|$ for IVM. On the other hand, even w.r.t. the induced test set

[29] A simple example is multi-class classification by using a C-process model with independent GP priors (see Section 2.1.2).

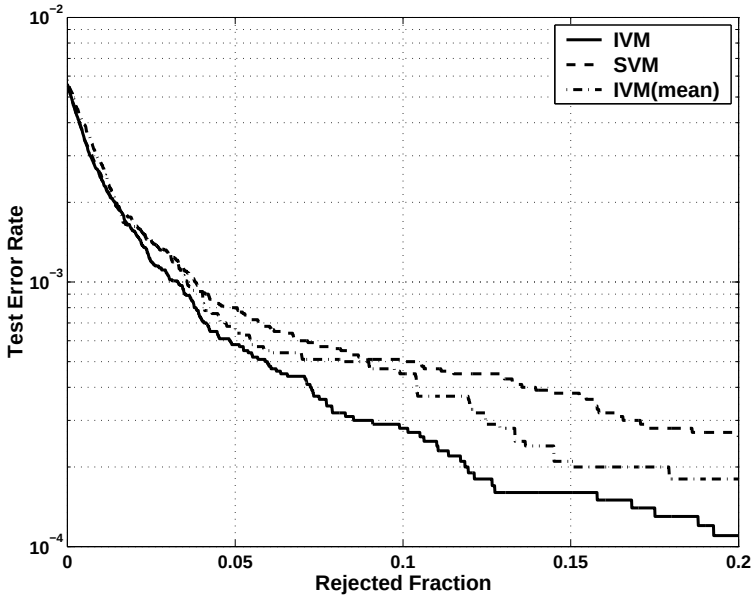

Figure 4.1: Test error rate against increasing rejection rate for MNIST, 9 vs. rest (averaged over 10 runs).

orderings they behave quite differently, giving no evidence for the usefulness of $s(\boldsymbol{x}_*)$ as estimate of $\mu(\boldsymbol{x}_*)$. The role of the predictive variances can be understood more clearly by plotting $|\mu(\boldsymbol{x}_*)|$ and $\sigma(\boldsymbol{x}_*)$ for misclassified points $\boldsymbol{x}_*$ only in the ordering given by $r(\boldsymbol{x}_*)$ (i.e. by the predictive probabilities), as is done (for one of the runs) in Figure 4.2.

We see that for this selection of points, large wrong values of $|\mu(\boldsymbol{x}_*)|$ are often accompanied by large values of $\sigma(\boldsymbol{x}_*)$. While the former alone lead to a suboptimal ordering w.r.t. the rejection scenario, their combination $r(\boldsymbol{x}_*)$ behaves better. Figure 4.3 shows the relationship between $\mu(\boldsymbol{x}_*)$ and $\sigma(\boldsymbol{x}_*)$ over the whole test set (we sorted the points w.r.t. their predictive mean value). The right hand plot shows mean $\pm$ one standard deviation computed by local averaging.[30]

The region $\mu(\boldsymbol{x}_*) \approx 0$ is sparsely populated due to the simplicity of the task. For $\mu(\boldsymbol{x}_*) \leq -5$ there is a noisy roughly linear relationship between predictive mean and standard deviation, but this does not hold in the area of real interest $\mu(\boldsymbol{x}_*) \approx 0$. Here, $\sigma(\boldsymbol{x}_*)$ fluctuates more strongly than for larger $|\mu(\boldsymbol{x}_*)|$ and we

[30]Using a box window of size 100. The run is the same as used for Figure 4.2.

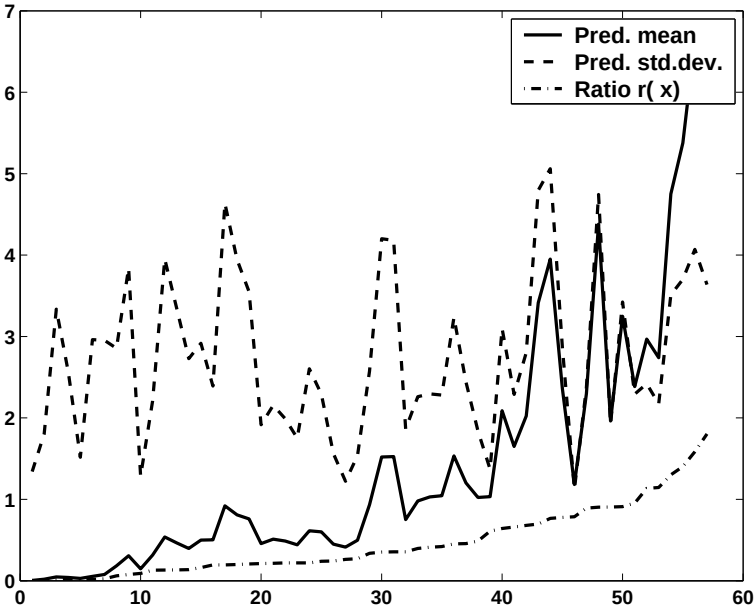

Figure 4.2: Absolute predictive mean $|\mu(\boldsymbol{x}_*)|$, standard deviation $\sigma(\boldsymbol{x}_*)$ and ratio $r(\boldsymbol{x}_*)$ for misclassified points (IVM, MNIST, 9 vs. rest).

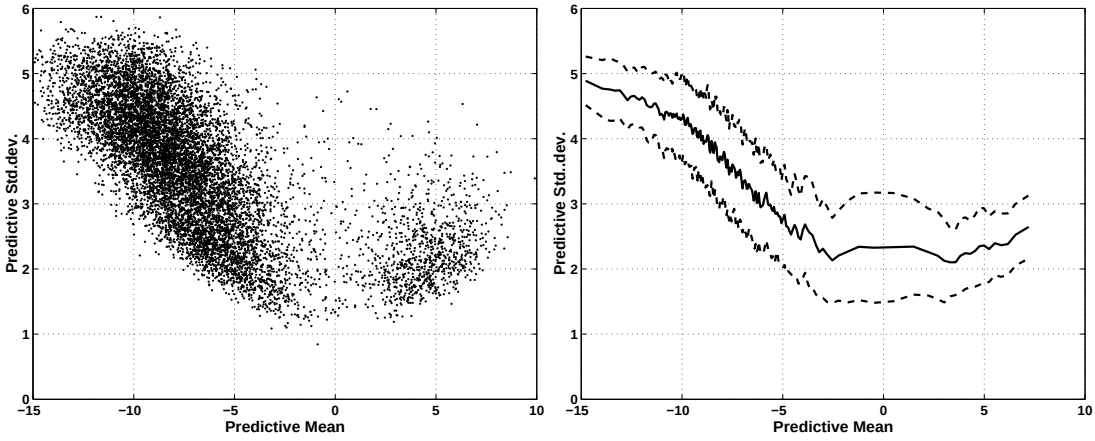

Figure 4.3: Predictive std.dev. against predictive mean over all test points (IVM, MNIST, 9 vs. rest). Right plot shows mean $\pm$ one standard deviation computed by local averaging.

have seen above that these fluctuations are not simply random, but systematically "correct" the predictive means w.r.t. ranking.

To conclude, we have shown that predicting posterior variances in an approxi-

mate Bayesian way does not only provide a more complete analysis by quantifying uncertainty, but can be quite essential for decision-theoretic setups beyond discrimination without a reject option. On the other hand, uncertainty estimates come at small but non-negligible extra cost when compared to optimised large margin discrimination methods.[31]

4.8 Experiments

In this section, we give results for a range of experiments testing the IVM and PLV schemes on real-world tasks.

4.8.1 IVM Classification: Digit Recognition

The material in the first part of this section has previously appeared in [98]. We used the MNIST handwritten digits database[32], comparing the IVM (see Section 4.4.1) against the SVM algorithm (see Section 2.1.6). We considered unbalanced binary tasks of the form 'c-against-rest', $c \in \{0, \ldots, 9\}$. c is mapped to $+1$, all others to -1. We down-sampled the bitmaps to size 13×13 and split the MNIST training set into a (new) training set of size $n = 59000$ and a validation set of size 1000; the test set size is 10000. A run consisted of model selection, training and testing, and all results are averaged over 10 runs. We employed the RBF (Gaussian) kernel with variance parameter $C > 0$ and an inverse squared length scale $w > 0$ (2.29). Model selection was done by minimising validation set error, training on random training set subsets of size 5000. [33]

Our goal was to compare the methods not only w.r.t. performance, but also running time. For the SVM, we chose the SMO algorithm [142] together with a fast elaborate kernel matrix cache (see [173] for details). For the IVM, we

[31] Platt's post-processing method requires to train SVM several times, leave data out for validation, etc., so the total cost is also quite a bit higher than simply training an SVM.

[32] Available online at *http://www.research.att.com/~yann/exdb/mnist/index.html*.

[33] The model selection training set for a run i is the same across tested methods. The list of kernel parameters considered for selection has the same size across methods.

employed randomised greedy selections with fairly conservative settings.[34] Since each binary digit classification task is very unbalanced, the bias parameter b in the GPC model was chosen to be non-zero. We simply fixed $b = \Phi^{-1}(r)$, where r is fraction of $+1$ patterns in the training set and added a constant $v_b = 1/10$ to the kernel K to account for the prior variance of the bias hyper-parameter. Ideally, both b and v_b should be chosen by model selection, but initial experiments with different values for (b, v_b) exhibited no significant fluctuations in validation errors. To ensure a fair comparison, we did initial SVM runs and initialised the active set size d with the average number of SVs found, independently for each c. We then re-ran the SVM experiments, allowing for $O(d\,n)$ cache space. Table 4.1 shows the results.

SVM				IVM			
c	d	gen	time	c	d	gen	time
0	1247	0.22	1281	0	1130	**0.18**	**627**
1	798	**0.20**	864	1	820	0.26	**427**
2	2240	**0.40**	2977	2	2150	**0.40**	**1690**
3	2610	0.41	3687	3	2500	**0.39**	**2191**
4	1826	0.40	2442	4	1740	**0.33**	**1210**
5	2306	**0.29**	2771	5	2200	0.32	**1758**
6	1331	**0.28**	1520	6	1270	0.29	**765**
7	1759	0.54	2251	7	1660	**0.51**	**1110**
8	2636	**0.50**	3909	8	2470	0.53	**2024**
9	2731	0.58	3469	9	2740	**0.55**	**2444**

Table 4.1: Test error rates (gen, %) and training times (time, s) on binary MNIST tasks. SVM: Support vector machine (SMO); d: average number of SVs. IVM: Sparse GPC, randomised greedy selections; d: final active set size. Figures are means over 10 runs.

Note that IVM shows comparable performance to the SVM, while achieving

[34] First 2 selections at random, then 198 using full greedy, after that a selection index of size 500 and a retained fraction $\tau = 1/2$.

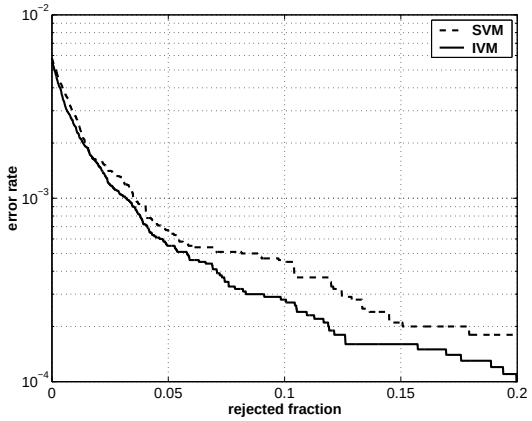

Figure 4.4: Test error rate against increasing rejection rate for the SVM (dashed) and IVM (solid), for the task $c = 9$ vs. the rest. For SVM, we reject based on "distance" from separating plane, for IVM based on estimates of predictive probabilities. The IVM line runs below the SVM line exhibiting lower classification errors for identical rejection rates.

lower training times. For less conservative settings of the randomised selection parameters, further speed-ups are realisable. We also registered (not shown here) significant fluctuations in training time for the SVM runs, while this figure is stable and a-priori predictable for the IVM. Within the IVM, we can obtain estimates of predictive probabilities for test points, quantifying prediction uncertainties. In Figure 4.4, which was produced for the hardest task $c = 9$, we reject fractions of test set examples based on the size of $|P(y_* = +1) - 1/2|$. For the SVM, the size of the discriminant output is often used to quantify predictive uncertainty heuristically. For $c = 9$, the latter is clearly inferior (the difference is less pronounced for the simpler binary tasks).

In the SVM community it is common to combine the 'c-against-rest' classifiers to obtain a multi-class discriminant as follows: for a test point $\boldsymbol{x}_*$, decide for the class whose associated classifier has the highest real-valued output. For the IVM, the equivalent would be to compare the estimates $\log P(y_* = +1)$ from each c-predictor and pick the maximising c. This is suboptimal, because the different predictors have not been trained jointly, thus constraints which have to hold between the latent processes and noise model parameters in order to obtain

a consistent distribution over classes are not enforced (the joint model is not properly calibrated). However, the IVM estimates of $\log P(y_* = +1)$ *do* depend on *predictive variances*, *i.e.* a measure of uncertainty about the predictive mean, which cannot be properly obtained within the SVM framework. If we use the probit noise model (as done here), $\log P(y_* = 1)$ lies between $\Phi(\mu_* + b)$ and $1/2$, depending on the value of the variance σ_*^2, which shows that the class with the largest $\log P(y_* = 1)$ need not be the one with the largest value for $\mu_* + b$. These points are discussed in more detail in Section 4.7.2. The combination scheme results in test errors of $1.54\%(\pm 0.0417\%)$ for IVM, $1.62\%(\pm 0.0316\%)$ for SVM (the figures are mean and standard deviation over 10 runs). When comparing these results to others in the literature, recall that our experiments were based on images sub-sampled to size 13×13 rather than the usual 28×28.

We have repeated the hardest of the tasks above ($c = 5, 8, 9$) using the full MNIST 28×28 input images together with a kernel implementation which exploits the high ratio of zero-valued pixels in these bitmaps. Apart from these changes, the setup is the same as above. We compared SVM against four variants of IVM which differed in the parameters for the randomised greedy selection mechanism (d_{full}: size of I until full greedy selection is run; $|J|$: size of selection set thereafter; τ: retain fraction): IVM-1: $d_{full} = 100$, $|J| = 300$, $\tau = 3/4$. IVM-2: $d_{full} = 50$, $|J| = 200$, $\tau = 3/4$. IVM-3: $d_{full} = 100$, $|J| = 500$, $\tau = 3/4$. IVM-4: $d_{full} = 100$, $|J| = 300$, $\tau = 1/2$. The results are shown in Table 4.2.

On the harder tasks SVM slightly outperforms the IVM variants. On the other hand, the nature of the IVM algorithm allows for a *a priori* prediction of required running time depending on the parameters d and the ones controlling RGS.

4.8.2 PLV Regression: Robot Arm Dynamics

The material in this section has previously appeared in [165]. The aim was to study the behaviour of PLV for GP regression with Gaussian noise (variance σ^2, a hyperparameter), especially w.r.t. model selection by marginal likelihood maximisation as discussed in Section 4.5.1. Note that we employ the approximation

	SVM	IVM-1	IVM-2	IVM-3	IVM-4
5 versus non-5					
gen(%)	0.29	0.33	0.33	0.30	0.31
time(s)	3363	2417	2920	3292	3070
d	3004	3200	3200	3200	3200
8 versus non-8					
gen(%)	0.44	0.51	0.51	0.49	0.50
time(s)	4338	3155	3353	3755	4150
d	3776	3600	3600	3600	3700
9 versus non-9					
gen(%)	0.55	0.65	0.62	0.62	0.63
time(s)	4282	2805	3444	4494	4510
d	3926	3600	3600	3900	3900

Table 4.2: Comparison of SVM vs. IVM on MNIST tasks (full-size images). Note that memory usage was limited to a maximum of $n \times 3600$, n the training set size.

(4.18) here, not the more general variational $\mathcal{G}$. Recall from Section 4.4.2 that in the special case of a Gaussian likelihood, the PLV scheme simplifies considerably, because the site parameters are fixed from the beginning and do not have to be adapted (since the true full posterior, conditioned on the hyperparameters is Gaussian, an EP approximation is not required). For these experiments, we also used the simple form of the approximate information gain selection score (see Section 4.4.2.1) which ignores all couplings (i.e., $H = \emptyset$). This version will be called *info-gain* here. We compared against a version which selects I at random (*random*) and against the method of Smola and Bartlett (*smo-bart*, see Section 4.6). For the latter, we deviated from the author's suggestions and choose a smaller random selection set size of $k = 30$. Note that *a priori*, *info-gain* and *random* have the same complexity, while *smo-bart* should be k times slower.

We chose the datasets *kin-40k* (10000 training, 30000 test cases, 9 attributes)

and *pumadyn-32nm* (7168 training, 1024 test cases, 33 attributes),[35] both artificially created using a robot arm simulator, highly non-linear and low-noise. We pre-processed both by subtracting off a linear regression fit and normalising all attributes to unit variance. Thus, a linear regression on these tasks would result in averaged squared errors ≈ 0.5. A better approach would use a semiparametric model (see Sections 2.1.5 and 2.1.7), but this has not been tried. In both cases, we used 10 different random splits into training and test sets, one for each run. For a fixed run, we used the same split over all experiments below, in order to facilitate cross-comparisons.

We used the *squared-exponential kernel* (2.30) with positive hyperparameters $\boldsymbol{W} = \mathrm{diag}(w_j)_j$, C, v_b and the hyperpriors employed by [146]. If model selection with this kernel is successful, it will "switch off" input attributes which are not relevant for inference by driving $w_j \to 0$ (*automatic relevance determination (ARD)*, see Section 2.1.8). Where not stated otherwise, experiments are repeated ten times, and we quote medians (as well as quartiles in the plots). Predictive accuracy is measured in terms of average squared error, i.e. $(1/2)(y_* - \mu(\boldsymbol{x}_*))^2$ averaged over the test set, note that this does not depend on the estimates of the predictive variance.

4.8.2.1 Learning Curves

Here, we compare full GPR against *info-gain*, *random*, *smo-bart*. Hyperparameters were adjusted by maximising their (marginal) log posterior for full GPR over a random subset of size n_{MS} of the training set. Figure 4.5 shows learning curves for kin-40k, $n_{MS} = 2000$ (note that each plot contains upper and lower quartiles for full GPR as horizontal lines) and Table 4.3 gives the corresponding training times.

For small sizes $d = |I|$, *smo-bart* outperforms the other methods, while for larger d, *random* seems less effective. We also see (from curves on the right) that sparse methods on the full training set ($n = 10000$) can significantly outperform

[35] *kin-40k*: thanks to Anton Schwaighofer (*www.igi.tugraz.at/aschwaig/data.html*). *pumadyn-32nm*: thanks to Zoubin Ghahramani (*www.cs.toronto/∼delve*).

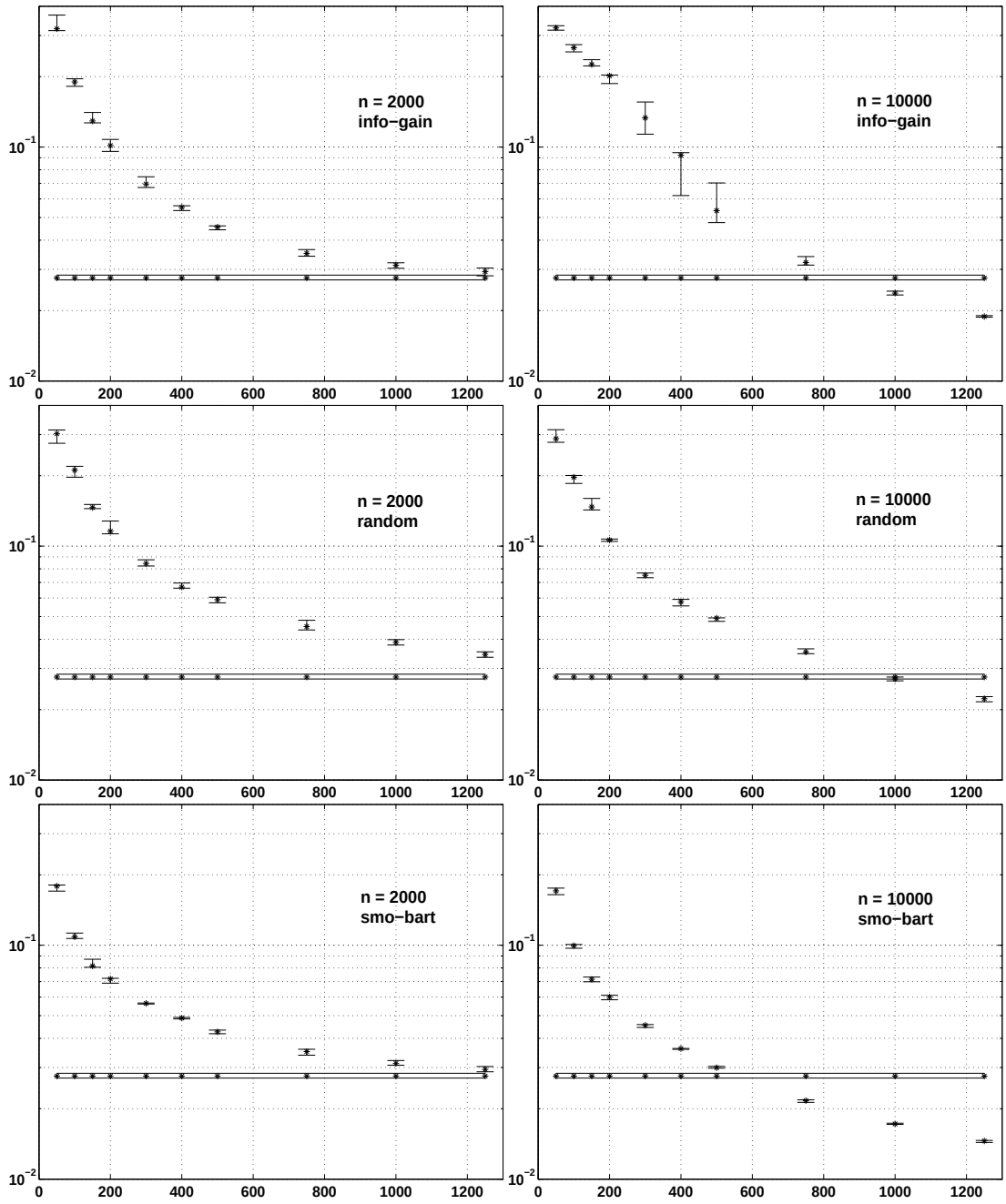

Figure 4.5: Learning curves for sparse GPR on kin-40k. Left: sparse methods use model selection training sets of full GPR, $n = 2000$. Right: sparse methods use full training set, $n = 10000$. X-axis: Active set size d. Y-axis: Average squared error. Horizontal lines: quartiles for full GPR, $n = 2000$.

	Time (secs) for act. set size d						
	100	200	300	400	500	1000	1250
	kin-40k, $n = 2000$						
info-gain	1.0	2.0	5.0	9.0	13.5	53.5	85.0
random	1.0	2.0	4.0	7.0	11.0	48.0	76.0
smo-bart	17.0	54.0	110.0	185.0	281.0	1088.0	1714.5
	kin-40k, $n = 10000$						
info-gain	5.0	13.0	25.5	42.0	62.5	230.0	352.0
random	3.0	10.0	21.0	35.5	55.0	215.0	338.5
smo-bart	88.0	265.0	530.5	885.0	1327.5	4977.0	7794.5

Table 4.3: Training time (secs) for methods *info-gain*, *random*, *smo-bart*.

full GPR on a subset ($n = 2000$), even though the former employ sparser expansions ($d = 1000, 1250$). In this case, the freedom of selecting amongst more points outweighs the advantage of using a larger expansion. While *random* and *info-gain* have similar training times, *smo-bart* is about thirty times slower, which probably precludes the latter being used as a subroutine for model selection. It would be fair to compare *info-gain* and *smo-bart* for equal amounts of running time, in which case the former performed much better.

The set *pumadyn-32nm* has a lot of irrelevant attributes which, in the context of GPR have to be identified in order to achieve good performance. Adapting the hyperparameters for $n_{MS} = 1024$ using full GPR leads to four attributes being singled out, all other $w_j \approx 0$. The median of the average squared test errors is 0.0225. Note that traditional techniques like cross-validation are not applicable in this situation, due to the large number (35) of hyperparameters. To demonstrate the significance of ARD, we ran the same experiment using the Gaussian (RBF) kernel (2.29), resulting in a squared error of 0.4730. However, if only the four "relevant" attributes are used, full GPR with the RBF kernel attains a squared error of 0.024. We see that covariance functions with many hyperparameters can be essential for decent performance, which stresses the importance of model selection techniques which can handle such covariance functions. Figure 4.6 shows

learning curves for the sparse methods.

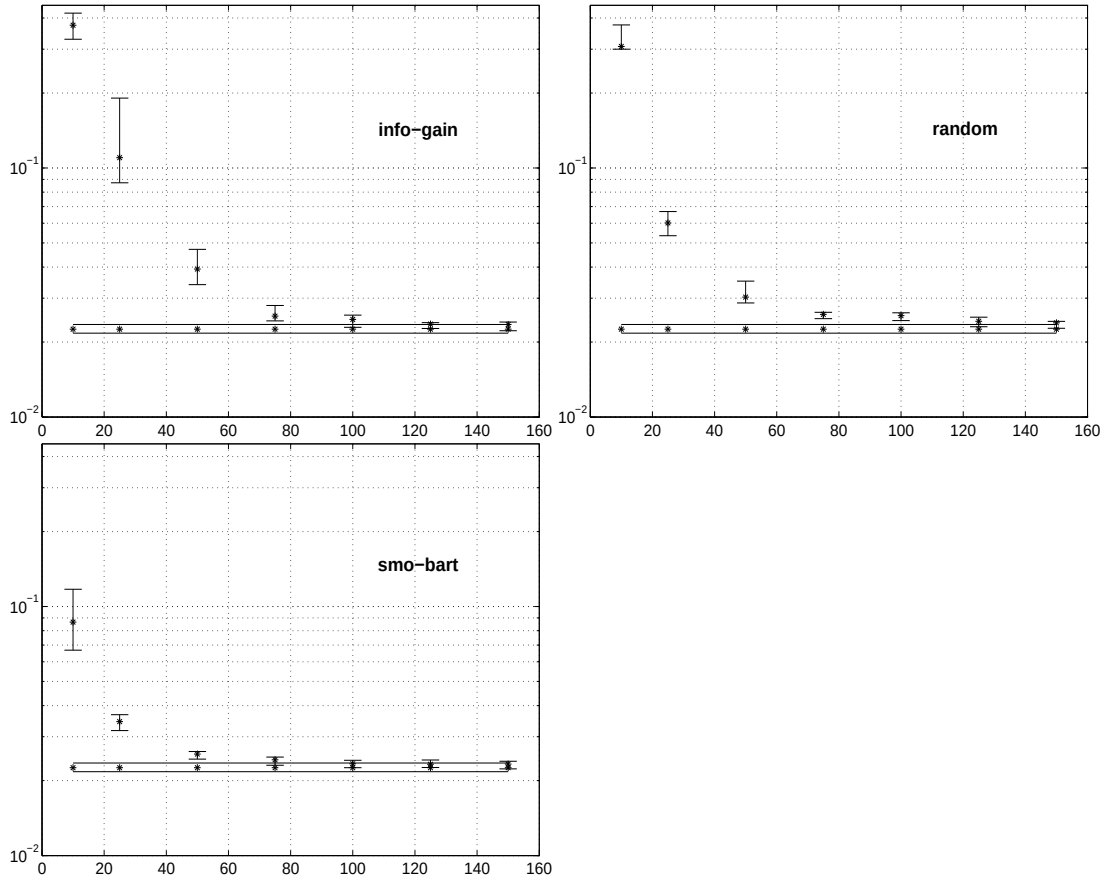

Figure 4.6: Learning curves for sparse GPR on pumadyn-32nm. X-axis: Active set size d. Y-axis: Average squared error.

Here, all three methods achieve an accuracy comparable to full GPR after the inclusion of $d = 125$ of the $n = 7168$ training cases, but while the training time median for full GPR is $1102.5s$, *info-gain*, *random* and *smo-bart* require $3s, 2s$ and $66s$ respectively. For unreasonably small values of d, *info-gain* exhibits some fluctuations, leading to a worse performance than *random*. The fact that *smo-bart* is 30 times slower than *info-gain* should be weighted against the slightly faster decay of its learning curve.

4.8.2.2 Sparse Model Selection

Here, we test selection of σ^2 and the kernel parameters by maximising the approximation of the (marginal) hyperposterior described in Section 4.4.2.1. *smo-bart* is not considered here due to excessive running time. In a first experiment, we followed the trajectory in hyperparameter space used by full GPR training (on pumadyn-32nm, $n_{MS} = 1024$) and computed the corresponding approximate criterion values for *info-gain*, *random* for different d. Figure 4.7 shows the criterion curves for *info-gain* (left) and *random* (right).

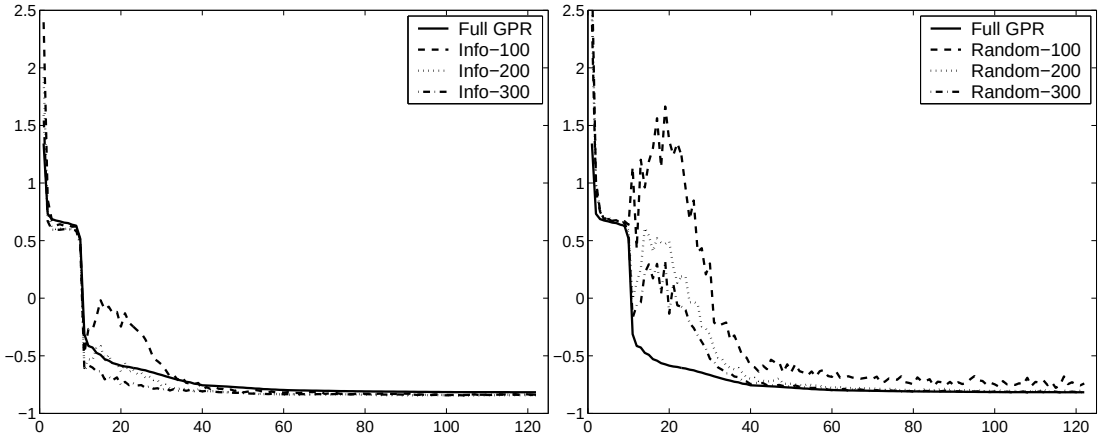

Figure 4.7: Criterion for full GPR and sparse approximations (evaluated at the same hyperparameters). X-axis: Iterations.

Along critical parts of the trajectory, the approximation given by *random* is fairly poor (for $d = 100, 200$), while the *info-gain* approximation seems acceptable. Note that for both methods, I is re-selected in every iteration, therefore the poor approximation by *random* cannot be attributed to random fluctuations.

In a larger experiment, we first trained full GPR on pumadyn-32nm, using $n_{MS} = 2048$. We then ran an identical setup,[36] however using the sparse approximate criterion together with *info-gain*, *random* and a variant of the latter, named *fixed* in which we selected I at random only once and kept it fixed during the

[36] The initial hyperparameter values were chosen as recommended in [146]. They are not close to a useful minimum point for this task (note the sharp initial drop of the criterion values). The optimiser was stopped once the relative improvement and gradient size fell below small thresholds, or after 200 iterations.

optimisation. We used active set sizes $d = 50, 100, 200, 500$. Only two different profiles were observed: either, the relevant attributes mentioned above were singled out (ARD) and the squared error was below 0.03, or the method failed with error close to 0.5. In fact, *random* and *fixed* for $d = 50, 100, 200$ failed in all runs, and *info-gain* for $d = 50$ was successful just once. For all other combinations, medians of the average squared error, number of iterations and optimisation running time over the successful runs are given in Table 4.4.

Method	d	# succ. runs	Squared error	# Iter. optim.	MS time (secs)
full GPR	-	10	0.0236	200	22100.5
info-gain	500	10	0.0296	200	9079
info-gain	200	10	0.0259	177.5	1833.5
info-gain	100	9	0.0252	200	836
random	500	8	0.0244	95	4412
fixed	500	9	0.0239	200	9203

Table 4.4: Comparison of model selection for full and several sparse GPR methods, dataset pumadyn-32nm, $n_{MS} = 2048$.

While model selection based on *info-gain* was reliable even for small sizes $d = 100, 200$, the runs for *random* often converged (to high accuracy) to poor spurious minima of the criterion approximation. Even in the successful runs, flat plateaus are traversed before a new downwards direction is found (see Figure 4.8, left).[37] Somewhat surprisingly, *info-gain* with $d = 500$ results in larger squared errors than our scheme for smaller d. In Figure 4.8, we compare criterion curves for full GPR and sparse methods using a run in which all of them were successful. On the right, we plot the sizes of the largest inverse length scales $w_j^{1/2}$ in the hyperparameter vectors chosen by the different methods (recall that the positions of these within $\boldsymbol{W}$ were the same for all methods).

[37]Somewhat surprisingly, *random* does not behave better than *fixed* w.r.t. spurious converge (for $d = 500$, they both fail in the same run). We would have expected *random* to escape spurious minima more readily, because I is re-selected at random in every iteration.

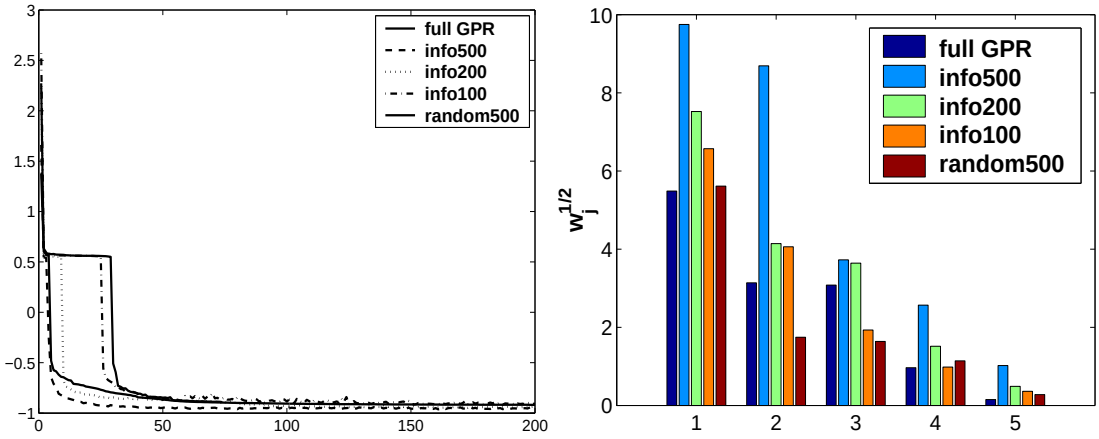

Figure 4.8: Left: Criterion curves for model selection optimisations (X-axis: Iterations of optimiser); lower solid line: full GPR; upper solid line: *random* ($d = 500$). Right: Largest values of inverse squared length scales w_j in the selected hyperparameter vector.

Note the dangerously flat plateau in the curve for *random* (*fixed* has a very similar curve, not shown here). As for the suboptimal performance of *info-gain* ($d = 500$), one may suspect overfitting[38] behaviour. The curve runs below the one for full GPR, and the concrete $w_j^{1/2}$ values are significantly larger than the ones chosen by the latter. Indeed, training full GPR using the hyperparameters found by *random*, $d = 500$, results in a squared error of 0.0317, suggesting that the hyperparameters are suboptimal.

4.8.3 IVM Regression: Robot Arm Dynamics

The aim of the empirical studies of this section is to test the model selection strategy for the IVM scheme proposed in Section 4.5.2 on the regression estimation task *pumadyn-32nm* already used in Section 4.8.2 and a direct comparison with model selection for the more expensive PLV scheme.

The setup is as follows. We use the same 10 dataset splits into $n = 7168$ training and $m = 1024$ test cases. As opposed to the PLV experiments where model selection (MS) was done on randomly chosen subsets of each training set

[38] In general, *empirical* Bayesian methods such as ours here can lead to overfitting.

of size $n_{MS} = 2048$, here we employ the full training sets for MS, $n_{MS} = n = 7168$. We employed the squared-exponential covariance function with the same hyperpriors and initial values as above. Recall from Sections 4.5.2 and 4.4.1 that a few additional parameters have to be specified for the IVM scheme. $|L|$ denotes the size of the MS criterion index L used to approximate the full likelihood in the MS criterion. Randomised greedy selection (RGS) comes with several parameters: d_{full}, $|J|$, τ (see Section 4.8.1). Note that the cost for a criterion and gradient computation is $O(|L|\, d^2)$ (see Section C.3.3).

Recall from Section 4.8.2.2 that for a decent performance on *pumadyn-32nm* with GPR, the discovery of "relevant" input dimensions is essential. In a first set of experiments, we asked whether the ARD effect can be reproduced reliably for model selection with the faster IVM scheme. We concentrated on the cases $d = 200$ and $d = 500$. For $d = 200$, we used $|L| = 4096$ (4/7 of the full training set) and did not employ RGS, while for $d = 500$, we used $|L| = 2048$, $d_{full} = 200$, $|J| = 1000$, $\tau = 1/2$. The (Quasi-Newton) optimiser was run using the same parameters as above, except that the maximum number of iterations for $d = 500$ was limited to 100 here (for $d = 200$, we adopted the very conservative bound of 200 from above). Again, no effort was made to choose a stopping criterion suitable to deal with the final oscillatory behaviour of the optimiser (see remarks in Section 4.5.3): we stopped once the relative improvement between adjacent steps fell below 10^{-5} or once the iteration bound was reached.

The "bimodal" behaviour in terms of predictive squared error was observed here as well. All ten runs for $d = 200$ failed to detect the relevant dimensions and had squared errors around 1/2. Each of these runs terminated prematurely, due to small relative improvements. The "solution" provided by these runs is invariably to drive C to a small value and σ^2 to a value around 1, producing predictive means and variances quite close to 0 respectively. This is close to the result linear regression would obtain (which is constant 0). We repeated these experiments with lower accuracy thresholds, forcing the optimiser to run the full 200 iterations, with no qualitative difference in the outcomes.

The situation was different for $d = 500$, for which all ten runs were successful

in detecting the four relevant components. The median averaged squared error was 0.0233, the median MS running time was 6519 secs. These figures should be compared against *info-gain*-(500) in table 4.4, keeping in mind that the latter ran for twice the number of optimiser iterations. IVM slightly outperformed all variants listed in table 4.4, but also runs somewhat longer than PLV for $d = 500$. Note that IVM operates on a larger training set (7192 vs. 2048 for PLV), but uses RGS and a MS selection index of size 2048 to avoid $O(n\,d^2)$ scaling. We repeated the same experiment with a less conservative RGS setting ($d_{full} = 50$, $|J| = 250$, $\tau = 3/4$), ending up with median squared error 0.0234 (again all runs were successful) and median MS time 5517 secs, a significant speed-up with no decrease in performance. We also ran IVM with full greedy selection on the same MS training subsets ($n_{MS} = 2048$) as used for the PLV experiments above. Here, three of the ten runs failed, one had squared error 0.0445, and the median over the remaining 6 runs was 0.0242, the median MS time was 4452 secs. Of course, just as in the PLV experiments, the final training run with the selected parameters was done on the full training set of 7192 patterns. We see that IVM with fairly aggressive RGS run on a large training set can be a much more robust alternative to subsampling the training set for model selection.

To conclude, on this particular task IVM regression with MS on the full training set slightly outperformed PLV regression with MS on a subset of the training set, and the running times are comparable. Since IVM with RGS is somewhat more complicated (especially w.r.t. bookkeeping), a larger gain could probably be realised by a more careful implementation. It should be kept in mind that for IVM, running time and complexity of implementation is the same for simple GP regression (Gaussian noise) and non-Gaussian models like binary classification (see Section 4.8.4), while PLV becomes much more complicated and costly in the latter case. On the other hand, the more accurate likelihood approximation of PLV allows us to use smaller active set sizes d before the inference approximation becomes too coarse to drive model selection: the *pumadyn-32nm* regression task is solved successfully for PLV with $d = 200$ and even $d = 100$ (see Section 4.8.2) while we required $d = 500$ for IVM here. The direct comparison between IVM

and PLV presented here is complicated further by the fact that we used different approximations for the marginal likelihood used to do model selection. For IVM, the variational lower bound from Section 4.5.2 was used, while for PLV we employed the simpler direct approximation (4.18). The latter is probably not sensible for IVM because it depends on the datapoints in I only.

4.8.4 IVM Classification: Digit Recognition II

In this section, we focus on model selection for IVM (see Section 4.5.2) on a binary classification task. The latter is extracted from the USPS handwritten digits database (see [99] or [161], Chap. A.1), containing grey-scale patterns of size 16×16, with attribute values quantised to five bits (values are $(u-15)/16$, $u = 0, \ldots, 30$, with $u = 0$ being attained for 43% of all values). The database comes split into a training set of $n = 7291$ and a test set of $m = 2007$ patterns. The class prior distributions are somewhat nonuniform, with 0(1194) and 1(1005) being most, 8(542) and 5(556) least represented. The corresponding recognition task is quoted to be rather hard, with human error rates estimated at around 2.5%.

Just as in Section 4.8.1 we employed the RBF kernel with variance parameter $C > 0$ and inverse squared length scale $w > 0$ (2.29). The hyperprior on $\log C$ was $N(-1, 1)$, on w^{-1} Gamma with mean 1, degrees of freedom 1. The probit noise model (2.9) had an unconstrained bias parameter b with a $N(\hat{b}, 5)$ hyperprior where $\hat{b}$ was computed based on the training data. We focussed on binary tasks of the form c-against-rest, $c \in \{0, \ldots, 9\}$.

For a first set of experiments, we employed a MS selection index size $|L| = 2000$ and RGS with parameters $d_{full} = 200$, $|J| = 250$, $\tau = 3/4$, furthermore an active set size $d = 500$. The results for all binary c-against-rest tasks are shown in Table 4.5.

Csató [38] quotes results of his sparse EP algorithm (which is a version of PLV with an on-line like selection of I) on USPS, 4-against-rest (Figure 5.6a in [38]) with test error ≈ 1.75, dropping to ≈ 1.65 for several sweeps (for $d = 500$). This is slightly worse than our result of 1.4, probably because the hyperparameters

c	gen	time	c	gen	time	c	gen	time
0	0.70	4514	1	0.70	5017	2	1.59	4419
3	1.30	4256	4	1.40	4419	5	1.25	4295
6	0.65	5017	7	0.65	4419	8	1.35	4256
9	0.80	4332						

Table 4.5: Test error rates (gen, %) and model selection times (time, s) on binary USPS tasks. Figures are medians over 10 runs.

(Csató uses the same covariance function and noise model) are set by hand in [38]. The curve in [38] suggests that our choice of $d = 500$ was very conservative.

We also combined the 10 binary predictors in the same heuristic manner as in Section 4.8.1 (deciding for the class c with maximum $\log P(y_* = +1)$ estimate), resulting in test error rate of 4.98%, comparable to results for other kernel classifiers on "black-box" covariance functions (see [161], Table 7.4). If we are allowed to reject an α-fraction of the test cases, it is sensible to avoid patterns for which the maximum over c of $\log P(y_* = +1)$ is smallest. Doing so, we obtain the error-reject curve in Figure 4.9. Csató [38] reports errors of 5.4% (single sweep) and 5.15% (three sweeps) on this task, however using $d = 350$ and subscribing to some restrictions we do not (for example, a single set I for all 10 classifiers is selected).

We also repeated the hardest binary task 2-against-rest using the squared-exponential covariance function (2.30). The median test error slightly dropped to 1.4% while the MS running time doubled to 8932 secs. All inverse squared length scales were between 2.4 and 5.8, thus no ARD effect was visible here: w.r.t. this particular kernel, none of the input components are irrelevant, or at least the MS method does not single out any. The sharp increase in running time is due to the $O(n\,d)$ part in the gradient term which has to be computed for each of the 259 hyperparameters.

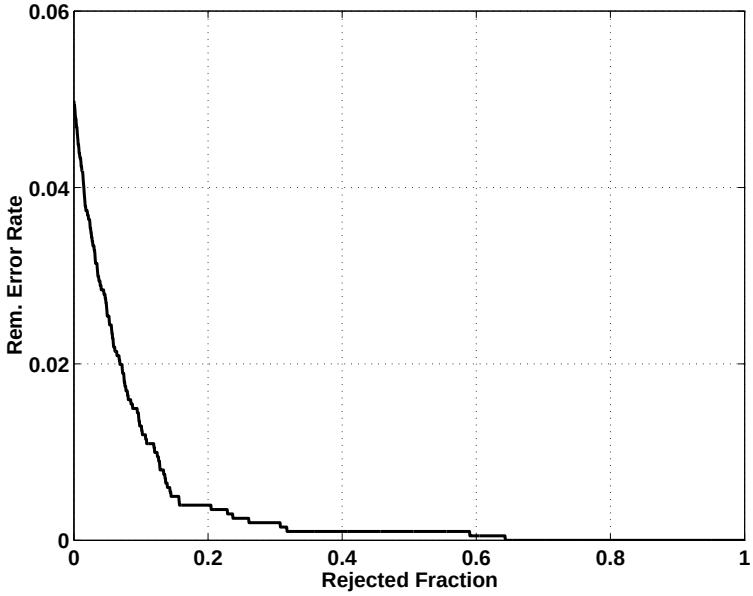

Figure 4.9: Test error rate against increasing rejection rate for IVM with model selection on combined USPS task. See text for details.

4.9 Discussion

In this chapter, we have discussed how to develop sparse approximations to Bayesian nonparametric GP methods in a principled manner. They lead to algorithms with training time scaling of $O(n\,d^2)$ and $O(d^2)$ per prediction, where d is a controllable parameter, and they "converge" towards their full Bayesian counterparts as $d \to n$. We have described two generic schemes to implement sparse greedy GP approximations based on greedy forward selection with information-theoretic criteria: the simple, efficient IVM (see Section 4.4.1) and the potentially more accurate PLV (see Section 4.4.2). For both schemes, we demonstrated how to do automatic model selection for a large number of hyperparameters by maximising sparse approximations to the marginal likelihood of the data, using modifications of standard optimisation techniques. Both schemes provide estimates of predictive variances along with predictive means, thus allowing for quantifying uncertainty in error bars or dealing with more general decision-theoretic setups than balanced discrimination. We established empirically that the methods presented in this chapter can be used to fit models involving priors with many

hyperparameters to large datasets from difficult real-world tasks.

Further conclusions and suggestions for future work can be found in Section 5.2.

Chapter 5

Conclusions and Future Work

In this final chapter, we present conclusions from the main contributions in this work, highlight unresolved issues and give some suggestions for valuable future work. This is done separately for the two main chapters in Section 5.1 (PAC-Bayesian theorems) and Section 5.2 (sparse GP approximations).

5.1 PAC-Bayesian Bounds for Gaussian Process Methods

In Chapter 3, we have given a simple and direct proof of a PAC-Bayesian generalisation error bound which generalises McAllester's original result. We have pointed out convex (Legendre) duality (see Section A.3) as the core technique in the proof, giving yet another example of the amazing scope and power of the duality principle in convex analysis (other applications relevant to machine learning include the EM algorithm, a number of variational approximations to Bayesian inference and convex (notably linear) programming). For certain classes of Bayes-type classifiers, convex duality provides a more direct (and therefore often tighter) approach to global (uniform) bounds than mathematically much more involved techniques based on combinatorial dimensions or covering numbers. We have shown how to apply PAC-Bayesian theorems to a large generic class of non-parametric Gaussian process methods spanning most schemes actually used in

practice, again without having to resort to "heavy-weight" mathematical concepts, ending up with practically tight results for powerful learning methods which are notoriously difficult to handle with conventional concepts. Although the PAC-Bayesian theorems certainly do not constitute the first application of convex duality for proving distribution-free bounds, we hope that the examples given in this thesis will spark new interest in establishing convex duality as core technique and developing its full potential within learning theory.

Our experimental results indicate that the PAC-Bayesian bounds can be very tight in practically relevant situations, giving more useful results than other state-of-the-art PAC bounds for kernel classifiers we considered. We have discussed possible reasons for lack of tightness of classical VC bounds in Section 3.1.1, and the same apply in principle to many of the current kernel classifier bounds we know of: the dependence on algorithm and data sample is typically very weak, given only through restricted statistics such as a large empirical margin or a certain degree of sparsity, and the bounds are hardly configurable by given task prior knowledge. In contrast, the PAC-Bayesian complexity measure, the relative entropy between posterior and prior, is more flexible, configurable and depends more strongly on the particular algorithm than any others employed in kernel classifier bounds we know of. In the GP case, prior knowledge can be encoded in a very general way via the choice of the covariance function.

Another "dual" relationship is worth pointing out. Approximate Bayesian techniques use simplifications to overcome the intractability of exact Bayesian analysis on a model, such as factorisation or Gaussian assumptions, and exactly these simplifications allow us to feasibly do a PAC analysis of the technique. The fact that we approximate the intractable posterior process by a Gaussian one allows us to compute the PAC-Bayesian bound for GP models. For a sparse GPC approximation, the computational complexity of evaluating the bound drops accordingly. Finally, relations between Gibbs and Bayes classifiers (see Section 3.2.5) can be inferred from symmetry properties of the approximate Gaussian predictive distribution, while they probably do not hold for the true predictive distribution. This example suggests that PAC or also average-case analyses could

be simplified (and tightened) if instead of exact (intractable) Bayesian inference we focussed on tractable approximations which are actually used in practice (another good example is given in [25]). Of course, analyses of the latter type are also of much higher interest to practitioners.

5.1.1 Suggestions for Future Work

The PAC-Bayesian theorem applies more widely to other approximate Bayesian inference techniques. For example, its application to parametric models is fairly straightforward if standard techniques like variational mean field or Laplace approximations are used. As a concrete example, it would be easy to apply it to Bayesian multi-layer perceptrons [110, 17], using inference approximations based on Laplace [110] or variational mean field [11]. Langford and Caruana [95] consider a different application to MLPs. Note that while the Gaussian approximations employed in approximate GP inference are rather natural (because of the GP prior) and often excellent, existing approximations to the posterior for complicated parametric models like MLPs can be very poor.[1] The theorem could also be applied to non-Bayesian methods resulting in classifiers which can be considered as probabilistic mixtures, such as the ones constructed by boosting methods [57, 59]. Gibbs versions of such classifiers would sample component classifiers using the mixture distribution for each prediction. Furthermore, the use of convex duality to decouple the posterior (with which the mixture classifier is defined) from the errors made by the individual component classifiers should have much wider applicability as a technique for proving PAC results for mixture classifiers (the mixtures could well be hierarchical, such as in the *mixture of experts* architecture [86, 205]).

Several open problems remain (to our knowledge). First, although Meir and Zhang [121] recently obtained a PAC-Bayesian theorem for Bayes-like classifiers (see Theorem 3.4), this result is typically less tight than the theorem for Gibbs classifiers used here, in apparent contradiction to the observation that in practice,

[1] This is often due to extreme multimodality of the posterior, together with other degeneracies such as plateaus, ridges, etc. In contrast to that, for many commonly used noise models, the GP posterior is unimodal and can be approximated very well by a Gaussian.

the Bayes version typically outperforms the Gibbs version (in our experiments, Theorem 3.4 could not give non-trivial guarantees). Refinements of this very recent result may lead to a more satisfying theorem for the Bayes version. As mentioned in Section 3.4.1.1, the problem is the scaling with the square root of $\mathrm{D}[Q \,\|\, P]/n$ together with the large leading constant. In the case of zero-one loss it is possible to derive variants of VC bounds which scale as $O(n^{-1})$ for small empirical error, but it is at least not straightforward to apply the same technique to the margin loss functions in Theorem 3.4. Second, a PAC-Bayesian theorem for restricted, yet unbounded loss functions would be required to deal with the regression estimation problem. Such a bound would necessarily depend on statistics of the data distribution, e.g. its variance, and would apply only if these statistics can somehow be controlled with certainty. Third, the PAC-Bayesian theorems do not allow for model selection based on the training sample S from a continuum of models and even the "stratification" procedures used to justify model selection from countable families seem unnatural compared with the elegant "union-bound free" approach of the PAC-Bayesian technique. The PAC-Bayesian theorem applies to hierarchical priors, so that integrating out hyperparameters (approximately) is admissible. However, empirical Bayesian model *selection* translates to integrating them out using spike approximate to the hyperposteriors, and this invalidates the bound because the relative entropy term becomes infinite.

5.2 Sparse Gaussian Process Methods

In Chapter 4, we have discussed how sparse approximations to Bayesian nonparametric GP methods can be developed in a principled manner. Our strict requirements on these approximations imply that conditional inference (training) scales as $O(n\,d^2)$ for running time and $O(n\,d)$ for memory[2], where the sparsity parameter d is controllable and can often be chosen to be much smaller than n with negligible loss of prediction performance. The time per prediction is in-

[2]The memory requirements can be reduced for the IVM scheme at the expense of slightly increased running time (see discussion of stub caching and trimming in Section C.3.1.1).

dependent of n and scales as $O(d^2)$. We placed special emphasis on solutions which are generic, can be implemented reasonably simply and in a numerically stable way without having to resort to complicated heuristics and which come with a favourable running time-memory trade-off, but at the same time provide the same range of "services" (e.g. predictive variances, model selection, *etc.*) as their non-sparse relatives.

We have shown that many sparse GP approximations can be understood naturally as marrying a conventional (non-sparse) technique with a suitable likelihood approximation which loosely speaking creates a bottleneck of a reduced number of "active" latent variables. Our aim was to design generic schemes which fit a large number of models of interest with minimal specific primitives to be added to an otherwise generic implementation. This was achieved by building on the EP algorithm which is generic up to simple computations involving low-dimensional Gaussian integrals over likelihood factors. The active set I, i.e. the subset of active latent variables upon which the likelihood approximation depends, is determined sequentially by greedy forward selection. More sophisticated and expensive evolutions of I (such as exchange moves) are suggested but not discussed in detail. The selection criteria we used (see Section 4.2.2) have been proposed for active learning (sequential design), but their application to sparse GP methods is novel (to our knowledge). In the context of the Gaussian approximations employed in Bayesian GP techniques, they can be computed easily and analytically, while their application to complicated nonlinear parametric architectures is much less attractive. Their computation is fast if the likelihood approximation factorises in a way similar to the true likelihood, and we proposed a simple "decoupling" criterion approximation which can be used if the likelihood approximation is fully coupled. Importantly, the methods we proposed are complete in that they include a generic empirical Bayesian model selection technique which is shown to work effectively in experiments with difficult large real-world tasks and heavily parameterised covariance functions, a setup which is very rarely considered in the kernel machines literature.

Not surprisingly, none of the basic ingredients used in our methods are en-

tirely new. Sparse approximations to nonparametric techniques, based on focussing on subsets of the dataset have been proposed many times, owing to the high significance in practice. The selection criteria we have employed here are well-established in active learning (sequential design). Finally, the sparse approximation of the GP expectation propagation scheme has been developed in [38], albeit with a different strategy for choosing I.

We should stress that our methods do not rely on special features of the covariance function or low input dimensionality, while the accuracy (and therefore usefulness) of the methods discussed here depends on how well the covariance matrix over the data can be approximated with a matrix of fairly low rank. It is our opinion that prior distributions (covariance functions in our case) should be chosen based on task prior knowledge with as few constraints for technical convenience as possible. An inference approximation should ideally be flexible in this respect, allowing for arbitrary covariance functions, noise models, and providing automatic model selection which scales linearly in the number of hyperparameters.

5.2.1 Suggestions for Future Work

The methods presented in Chapter 4 are generic and applicable to many nonparametric models beyond binary classification and regression with Gaussian noise. Robust regression, e.g. with t-distributed noise would be straightforward to do.[3] Multi-class classification or ordinal regression ("ranking") would require a C-process model (see Section 2.1.2), running EP with a likelihood which factors over groups of C latent variables each. This is implicit in our general development, with the caveat that the running time complexity increases by a factor of C^2. Further approximations to reduce this to a factor of C may be applicable (for example, the Laplace GP approximation of Section 2.1.3 scales linearly with C). Applications of nonparametric kernel methods are not restricted to standard supervised statistical tasks like classification or regression estimation.

[3] Our implementation allows to plug in arbitrary noise models, as long as certain primitives are implemented. It can also be used with arbitrary covariance functions.

Csató et. al. [41] use sparse GP techniques to approximate density estimation. Bach and Jordan [8] employ low-rank kernel matrix approximations in the context of approximations to the mutual information and apply their contrast function to independent components analysis. Friedman and Nachman [60] propose Gaussian process networks to model sets of random variables with complicated nonlinear interactions (and only vague prior knowledge about their characteristics) and show how to learn the network structure. It will be interesting to find out whether the generic sparse methods developed here are applicable to these more ambitious (and expensive) applications of nonparametric smoothing.

The code used for our experiments has not yet been released into the public domain, this is a pressing issue for future work. It is written in C++, easily extensible but unfortunately not as user-friendly as a Matlab implementation would be.[4] We decided against a Matlab implementation fairly early, because Matlab is known for its poor memory management and its inability of handling iterative code. A future project would be to wrap our code as MEX functions which could then be called from Matlab.

[4]It also contains some proprietary code (Numerical Recipes) in the moment which has to be replaced before even the present version can be released.

Appendix A

General Appendix

This chapter provides the general background which we draw from in the remainder of the thesis and is included mainly to render the work self-contained. Many readers will be familiar with most of the material and are invited consult this chapter only to match their notational conventions with ours.

Section A.1 defines general notational conventions. In Section A.2 we state properties from linear algebra and collect useful formulae. In Section A.3 we discuss properties of convex functions central to the first part of this thesis. Section A.4 introduces exponential families, in particular the Gaussian one and some of its properties relevant to this work. In Section A.5, we briefly define some notions of pattern recognition, the statistical problem this thesis is mostly concerned with. Bayesian inference and some approximations towards such is the topic of Section A.6. Finally, Section A.7 states basics about large deviation inequalities.

A.1 Notation

A.1.1 Linear Algebra

Vectors $\boldsymbol{a} = (a_i)_i = (a_1 \dots a_n)^T$ (column by default) and matrices $\boldsymbol{A} = (a_{i,j})_{i,j}$ are written in bold-face. If $\boldsymbol{A}$ is a matrix, $\boldsymbol{a}_j$ denotes its j-th column, $a_{i,j}$ its (i,j)-th entry. If $\boldsymbol{A} \in \mathbb{R}^{m,n}$, $I \subset \{1,\dots,m\}$, $J \subset \{1,\dots,n\}$ are index

sets,[1] then $\boldsymbol{A}_{I,J}$ denotes the $|I| \times |J|$ sub-matrix formed from $\boldsymbol{A}$ by selecting the corresponding entries (i, j), $i \in I$, $j \in J$. We use some short notations: $\boldsymbol{A}_{k\ldots l,J} = \boldsymbol{A}_{\{k\ldots l\},J}$, $\boldsymbol{A}_{i,J} = \boldsymbol{A}_{\{i\},J}$, $\boldsymbol{A}_{\cdot,J} = \boldsymbol{A}_{1\ldots m,J}$ and $\boldsymbol{A}_{\setminus I,J} = \boldsymbol{A}_{\{1\ldots m\}\setminus I,J}$, etc. Here, $\setminus I = \{1, \ldots, m\} \setminus I$ is the complement of I, sorted in ascending order. For example, $\boldsymbol{A}_{i,j} = a_{i,j}$, $\boldsymbol{A}_{\cdot,j} = \boldsymbol{a}_j$ and $\boldsymbol{A}_{\cdot,\cdot} = \boldsymbol{A}$.

Some special vectors and matrices are defined as follows: $\mathbf{0} = (0)_i$ and $\mathbf{1} = (1)_i$ the vectors of all zero and all ones, $\boldsymbol{\delta}_j = (\delta_{i,j})_i$ the j-th standard unit vector. Here, $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise (Kronecker symbol). Furthermore, $\boldsymbol{I} = (\delta_{i,j})_{i,j}$ is the identity matrix. For these matrices, the size will always be clear from the context.

The superscript T denotes transposition. $\operatorname{diag} \boldsymbol{a}$ is the matrix with diagonal $\boldsymbol{a}$ and 0 elsewhere. $\operatorname{diag} \boldsymbol{A}$ is the vector containing the diagonal of $\boldsymbol{A}$, and $\operatorname{diag}^2 \boldsymbol{A}$ is the diagonal matrix with the same diagonal as $\boldsymbol{A}$. $\operatorname{tr} \boldsymbol{A}$ is the sum of the diagonal elements of $\boldsymbol{A}$, $\operatorname{tr} \boldsymbol{A} = \mathbf{1}^T(\operatorname{diag} \boldsymbol{A})$. diag and tr operators have lower priority than multiplication. For example, $\operatorname{diag} \boldsymbol{A}^T\boldsymbol{b}$ is the diagonal matrix from $\boldsymbol{A}^T\boldsymbol{b}$, *not* the inner product of $\operatorname{diag} \boldsymbol{A}$ and $\boldsymbol{b}$. $|\boldsymbol{A}|$ denotes the determinant of the square matrix $\boldsymbol{A}$. For $p > 1$, $\|\boldsymbol{a}\|_p$ denotes the p-norm of the vector $\boldsymbol{a}$, $\|\boldsymbol{a}\|_p = (\sum_i |a_i|^p)^{1/p}$. If nothing else is said, $\|\cdot\| = \|\cdot\|_2$, the Euclidean norm. If $\boldsymbol{A}$, $\boldsymbol{B}$ have the same size, the Hadamard product (or direct product, or component-wise product) is defined as $\boldsymbol{A} \circ \boldsymbol{B} = (a_{i,j}b_{i,j})_{i,j}$. The symmetrisation function is $\operatorname{sym} \boldsymbol{A} = (1/2)(\boldsymbol{A} + \boldsymbol{A}^T)$. Relations are vectorised in Matlab style, as are scalar functions: $\boldsymbol{a} \geq \boldsymbol{b}$ means that $a_i \geq b_i$ for all i, and $f(\boldsymbol{a}) = (f(a_i))_i$.

A.1.2 Probability. Miscellaneous

We do not distinguish notationally between a random variable and its possible values. Vector or matrix random variables are written in the same way as vectors or matrices. If a distribution has a density, we generally use the same notation for the distribution and its density function. If $\boldsymbol{x}$ is a random variable, then $\mathrm{E}[\boldsymbol{x}]$ denotes the expectation (or expected value) of $\boldsymbol{x}$. If A is an event, then

[1] All index sets and sets of data points are assumed to be ordered, although we use a notation known from unordered sets.

$\Pr\{A\}$ denotes its probability. The probability space will usually be clear from the context, but for clarity we often use an additional subscript, e.g. $\Pr_S\{A\}$ or $\mathrm{E}_{\boldsymbol{x}\sim P}[\boldsymbol{x}]$. The latter is sometimes abbreviated to $\mathrm{E}_P[\boldsymbol{x}]$. By I_A, we denote the indicator function of an event A, i.e. $\mathrm{I}_A = 1$ if A is true, $\mathrm{I}_A = 0$ otherwise. Note that $\Pr\{A\} = \mathrm{E}[\mathrm{I}_A]$. The delta distribution $\delta_{\boldsymbol{x}}$ places mass 1 onto the point $\boldsymbol{x}$ and no mass elsewhere, $\delta_{\boldsymbol{x}}(B) = \mathrm{I}_{\{\boldsymbol{x}\in B\}}$. Let $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$ be sets of random variables, $\mathcal{X}$, $\mathcal{Y}$ non-empty. We write $\mathcal{X} \perp \mathcal{Y} \,|\, \mathcal{Z}$ to denote the conditional independence of $\mathcal{X}$ and $\mathcal{Y}$ given $\mathcal{Z}$: the conditional distribution of $\mathcal{X}$ given $\mathcal{Y}$, $\mathcal{Z}$ does not depend on $\mathcal{Y}$.

log denotes the logarithm to Euler's base e. The notation $f(\boldsymbol{x}) \propto g(\boldsymbol{x})$ means that $f(\boldsymbol{x}) = cg(\boldsymbol{x})$ for $c \neq 0$ constant w.r.t. $\boldsymbol{x}$. We often use this notation with the left hand side being a density. By $\operatorname{sgn} x$, we denote the sign of x, i.e. $\operatorname{sgn} x = +1$ for $x > 0$, $\operatorname{sgn} x = -1$ for $x < 0$, and $\operatorname{sgn} 0 = 0$. The Landau O-notation is defined as $g(n) = O(f(n))$ iff there exists a constant $c \geq 0$ such that $g(n) \leq c\, f(n)$ for almost all n.

We use some probability-theoretic concepts and notation which might be unfamiliar to the reader. A measure is denoted by $d\mu(\boldsymbol{x})$, the Lebesgue measure in $\mathbb{R}^d$ is denoted by $d\boldsymbol{x}$. If A is a measurable set, $\mu(A) = \int \mathrm{I}_{\{\boldsymbol{x}\in A\}} d\mu(\boldsymbol{x})$ denotes its mass under μ. A measure is finite if the mass of the whole space is finite, and a probability measure if this mass is 1. If $d\mu$ is a probability measure, we denote its distribution by μ. The sets A of mass 0 are called *null sets*.[2] $d\mu_1$ is called *absolutely continuous* w.r.t. $d\mu_2$ if all null sets of $d\mu_1$ are null sets of $d\mu_2$ (the notation is $d\mu_1 \ll d\mu_2$). The theorem of Radon and Nikodym states that $d\mu_1$ has a *density* $f(\boldsymbol{x})$ w.r.t. $d\mu_2$, i.e.

$$\mu_1(A) = \int \mathrm{I}_{\{\boldsymbol{x}\in A\}} f(\boldsymbol{x})\, d\mu_2(\boldsymbol{x})$$

for all $d\mu_1$-measurable A, iff $d\mu_1 \ll d\mu_2$. In this case,

$$f(\boldsymbol{x}) = \frac{d\mu_1(\boldsymbol{x})}{d\mu_2(\boldsymbol{x})}$$

is called *Radon-Nikodym derivative* or simply density w.r.t. $d\mu_2$.

[2] We always assume that the underlying probability space is complete, i.e. its σ-field contains all subsets of null sets.

A.2 Linear Algebra. Useful Formulae

A.2.1 Partitioned Matrix Inverses. Woodbury Formula. Schur Complements

Definition A.1 (Schur Complement) *Let $\boldsymbol{A} \in \mathbb{R}^{n,n}$ be some nonsingular matrix, and $I \subset \{1,\ldots,n\}$ an ordered index set, $I \neq \emptyset$, $I \neq \{1,\ldots,n\}$. Let $R = \{1,\ldots,n\} \setminus I$ be the complement of I, and assume that $\boldsymbol{A}_I$ is nonsingular as well*[3]*. The* Schur complement *$\boldsymbol{A}/\boldsymbol{A}_I$ is defined as*

$$\boldsymbol{A}/\boldsymbol{A}_I = \boldsymbol{A}_R - \boldsymbol{A}_{R,I}(\boldsymbol{A}_I)^{-1}\boldsymbol{A}_{I,R}.$$

Note that since both $\boldsymbol{A}$ and $\boldsymbol{A}_I$ are nonsingular, so is the Schur complement, and

$$|\boldsymbol{A}| = |\boldsymbol{A}/\boldsymbol{A}_I|\,|\boldsymbol{A}_I|\,.$$

If furthermore $\boldsymbol{A}_R$ is nonsingular, the same relation holds for $\boldsymbol{A}/\boldsymbol{A}_R$, and for the special case $\boldsymbol{A}_I = \boldsymbol{I},\ \boldsymbol{A}_R = \boldsymbol{I}$ (but of different size), we have the useful relation

$$\left|\boldsymbol{I} + \boldsymbol{U}\boldsymbol{U}^T\right| = \left|\boldsymbol{I} + \boldsymbol{U}^T\boldsymbol{U}\right|,$$

whenever the determinants exist.

Lemma A.1 (Partitioned Inverse Equations) *Assume that $\boldsymbol{A}_I$ is nonsingular. Then, $\boldsymbol{A}$ is nonsingular iff the Schur complement $\boldsymbol{A}/\boldsymbol{A}_I$ is invertible, and in this case:*

$$\begin{aligned}
\left(\boldsymbol{A}^{-1}\right)_R &= \left(\boldsymbol{A}/\boldsymbol{A}_I\right)^{-1}, \\
\left(\boldsymbol{A}^{-1}\right)_{I,R} &= -\left(\boldsymbol{A}/\boldsymbol{A}_I\right)^{-1}\left(\boldsymbol{A}_{R,I}(\boldsymbol{A}_I)^{-1}\right), \\
\left(\boldsymbol{A}^{-1}\right)_{R,I} &= -\left((\boldsymbol{A}_I)^{-1}\boldsymbol{A}_{I,R}\right)\left(\boldsymbol{A}/\boldsymbol{A}_I\right)^{-1}, \\
\left(\boldsymbol{A}^{-1}\right)_I &= (\boldsymbol{A}_I)^{-1} + \left((\boldsymbol{A}_I)^{-1}\boldsymbol{A}_{I,R}\right)\left(\boldsymbol{A}/\boldsymbol{A}_I\right)^{-1}\left(\boldsymbol{A}_{R,I}(\boldsymbol{A}_I)^{-1}\right).
\end{aligned}$$

Of course we can simply interchange I and R in these equations, which renders the *Woodbury formula* as a useful corollary.

[3] Generalisations are possible using pseudo-inverses, but we do not require them here.

Lemma A.2 (Woodbury Formula) *Assume that both $\boldsymbol{A}_I$, $\boldsymbol{A}_R$ are nonsingular. Then, $\boldsymbol{A}$ is nonsingular iff any of the Schur complements $\boldsymbol{A}/\boldsymbol{A}_I$, $\boldsymbol{A}/\boldsymbol{A}_R$ are invertible. In this case, both complements are nonsingular, and we have*

$$\begin{aligned}(\boldsymbol{A}/\boldsymbol{A}_R)^{-1} &= \left(\boldsymbol{A}_I - \boldsymbol{A}_{I,R}\boldsymbol{A}_R^{-1}\boldsymbol{A}_{R,I}\right)^{-1} \\ &= (\boldsymbol{A}_I)^{-1} + (\boldsymbol{A}_I)^{-1}\boldsymbol{A}_{I,R}\left(\boldsymbol{A}/\boldsymbol{A}_I\right)^{-1}\boldsymbol{A}_{R,I}(\boldsymbol{A}_I)^{-1},\end{aligned} \tag{A.1}$$

since both sides are equal to $(\boldsymbol{A}^{-1})_I$.

The Woodbury formula is most useful if $|R|$ is much smaller than $|I|$. Its straightforward application often leads to severe numerical instabilities, and it should be used as a symbolic rewriting tool rather than for doing actual computations. The incremental Cholesky decomposition (see Section A.2.2) should be preferred. For more details on Schur products, see [78].

A.2.2 Update of Cholesky Decomposition

In the context of linear systems with symmetric positive definite matrices, the Cholesky decomposition is the method of choice (for an exact treatment), due to its high numerical stability, efficiency and simplicity.

Definition A.2 (Cholesky Decomposition) *Let $\boldsymbol{M} \in \mathbb{R}^{n,n}$ be a symmetric matrix. The Cholesky decomposition of $\boldsymbol{M}$ exists iff $\boldsymbol{M}$ is positive definite, i.e. $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$. It is defined as*

$$\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T,$$

where the Cholesky factor $\boldsymbol{L}$ is lower triangular with positive elements on the diagonal.

$\boldsymbol{L}$ is defined uniquely by this relation. Computing the Cholesky factor is algorithmically straightforward, and in contrast to many other decompositions, pivoting and row/column permutations are not necessary. The decomposition costs $O(n^3)$, but is more than twice as fast as a matrix inversion.[4] The Cholesky factor can

[4] In fact, the proper and fastest way to invert $\boldsymbol{M}$ is to compute $\boldsymbol{L}$, then $\boldsymbol{M}^{-1}$ from the latter.

be stored together with the system matrix itself in *one* n-by-n matrix plus an additional n-vector. Once $\boldsymbol{L}$ is computed, systems $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}$ are solved by *back-substitution*: $\boldsymbol{L}\boldsymbol{u} = \boldsymbol{b}$, $\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{u}$. Furthermore, $\boldsymbol{a}^T\boldsymbol{M}^{-1}\boldsymbol{b} = (\boldsymbol{L}^{-1}\boldsymbol{a})^T(\boldsymbol{L}^{-1}\boldsymbol{b})$. Two back-substitutions come at the cost of one general matrix-vector multiplication.

The Woodbury formula (A.1), although often useful as a symbolic rewriting tool, is notoriously unstable in the presence of round-off errors. It is our opinion that for explicit computations, Cholesky techniques should *always* be preferred.[5] Here is how you update the Cholesky factor for rank-1 changes of the system matrix. In the sequel, we will use a slightly different variant of the Cholesky decomposition, namely let $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ where $\boldsymbol{D}$ is diagonal with positive entries and $\operatorname{diag}\boldsymbol{L} = \boldsymbol{I}$. Suppose we want to compute the decomposition for $\boldsymbol{M} + \boldsymbol{v}\boldsymbol{v}^T$. First, solve $\boldsymbol{L}\boldsymbol{p} = \boldsymbol{v}$ by back-substitution, thus $\boldsymbol{M} + \boldsymbol{v}\boldsymbol{v}^T = \boldsymbol{L}(\boldsymbol{D} + \boldsymbol{p}\boldsymbol{p}^T)\boldsymbol{L}^T$. If we find $\tilde{\boldsymbol{L}}\boldsymbol{D}'\tilde{\boldsymbol{L}}^T = \boldsymbol{D} + \boldsymbol{p}\boldsymbol{p}^T$, the new factor is $\boldsymbol{L}\tilde{\boldsymbol{L}}$ and the new diagonal matrix is $\boldsymbol{D}'$.

It turns out that $\tilde{\boldsymbol{L}}$ really only depends on $O(n)$ parameters and can be computed in $O(n)$ given $\boldsymbol{p}$. Namely, $\tilde{\boldsymbol{L}}_{i,j} = p_i\beta_j$, $i > j$ and $\tilde{\boldsymbol{L}}_{i,i} = 1$. One can also show that these parameters come out of the following recurrence: start with $t_0 = 1$, then iterate $i = 1, \ldots, n$:

$$a = t_{i-1}d_i + p_i^2, \quad t_i = \frac{a}{d_i}, \quad d_i' = \frac{a}{t_{i-1}}, \quad \beta_i = \frac{p_i}{a},$$

where β_n is not needed. $\boldsymbol{D}'$ can overwrite $\boldsymbol{D}$. If we want to update $\boldsymbol{M}$ to $\boldsymbol{M} - \boldsymbol{v}\boldsymbol{v}^T$, the same method applies (given that the resulting matrix is positive definite), but the following recurrence should be used instead: start with $s_0 = -1$, then iterate $i = 1, \ldots, n$:

$$a = p_i s_{i-1}, \quad d_i' = d_i + p_i a, \quad \beta_i = \frac{a}{d_i'}, \quad s_i = s_i - a\beta_i.$$

Due to the simple form of $\tilde{\boldsymbol{L}}$, back-substitutions or multiplications with this factor can be done in $O(n)$. For example, if matrices of the form $\boldsymbol{L}^{-1}\boldsymbol{A}$ are to

[5] Unfortunately, these techniques are only poorly supported in standard Matlab. The command `cholupdate` performs a rank-1 update, but returns the new Cholesky factor explicitly, instead of allowing for the implicit $\tilde{\boldsymbol{L}}$ form.

be updated in the sense that $\boldsymbol{L}$ is replaced by the new Cholesky factor, this can be done efficiently as $(\boldsymbol{L}\tilde{\boldsymbol{L}})^{-1}\boldsymbol{A} = \tilde{\boldsymbol{L}}^{-1}(\boldsymbol{L}^{-1}\boldsymbol{A})$. The modifications for the case $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T$ are obvious: replace $\boldsymbol{D}$ by $\boldsymbol{I}$ and right-multiply $\tilde{\boldsymbol{L}}$ by $(\boldsymbol{D}')^{1/2}$.

In order to physically update the Cholesky factor $\boldsymbol{L}$ we can use the in-place scheme described in Algorithm 3. Again, if $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T$, $\boldsymbol{L}'$ has to be right-multiplied by $(\boldsymbol{D}')^{1/2}$ afterwards. Although this technique is routinely used in numerical programming, we have come across it via [54].

Algorithm 3 Update of Cholesky Factor

for $i = 1, \ldots, n$ **do**
  $\rho = l_{i,i}p_i,\ \sigma = 0.\ l'_{i,i} = l_{i,i}.$
  for $j = i-1, i-2, \ldots, 1$ **do**
    $\sigma \leftarrow \sigma + \rho.\ \rho = l_{i,j}p_j.$
    $l'_{i,j} = l_{i,j} + \sigma\beta_j$
  end for
end for

A.2.3 Some Useful Formulae

In this section, we collect some formulae which do not fit in elsewhere.

Lemma A.3 (Tower Properties) *Let* $\boldsymbol{x}$, $\boldsymbol{y}$ *be random variables such that* $\mathrm{E}[\boldsymbol{y}\boldsymbol{y}^T]$ *exists.*

$$\begin{aligned} \mathrm{E}[\boldsymbol{y}] &= \mathrm{E}\left[\mathrm{E}[\boldsymbol{y} \mid \boldsymbol{x}]\right], \\ \mathrm{Var}[\boldsymbol{y}] &= \mathrm{Var}\left[\mathrm{E}[\boldsymbol{y} \mid \boldsymbol{x}]\right] + \mathrm{E}\left[\mathrm{Var}[\boldsymbol{y} \mid \boldsymbol{x}]\right]. \end{aligned} \tag{A.2}$$

Here, $\mathrm{Var}[\boldsymbol{y}] = \mathrm{E}[(\boldsymbol{y} - \mathrm{E}[\boldsymbol{y}])(\boldsymbol{y} - \mathrm{E}[\boldsymbol{y}])^T]$ *and* $\mathrm{Var}[\boldsymbol{y} \mid \boldsymbol{x}] = \mathrm{E}[(\boldsymbol{y} - \mathrm{E}[\boldsymbol{y} \mid \boldsymbol{x}])(\boldsymbol{y} - \mathrm{E}[\boldsymbol{y} \mid \boldsymbol{x}])^T \mid \boldsymbol{x}]$.

A.3 Convex Functions

Properties of convex functions and sets are of central importance in this thesis and have an immense number of applications in machine learning and statistics.

A convex function can be globally lower-bounded by a hyperplane, and the bound is locally tight wherever desired. Some of the powerful consequences of this simple fact are developed here, a comprehensive treatment of the subject can be found in [152]. The literature on convex optimisation is large, we refer to [24, 76, 77].

Definition A.3 (Convexity) *A subset $\mathcal{C} \subset \mathcal{V}$ of a vector space $\mathcal{V}$ is* convex *if for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}$, $\lambda \in (0,1)$, $\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \mathcal{C}$.*
A function $f : \mathcal{C} \to \mathbb{R}$ ($\mathcal{C}$ convex) is convex *if for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{C}$, $\lambda \in (0,1)$,*

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2).$$

Note that this is equivalent to the set $\mathrm{epi}(f) = \{(\boldsymbol{x}, y) \in \mathcal{C} \times \mathbb{R} \,|\, f(\boldsymbol{x}) \leq y\}$ *being convex. f is called* strictly convex *if the inequality is strict whenever $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$.*

Note that if $\mathcal{C}$ is open and f is twice continuously differentiable on $\mathcal{C}$ then f is strictly convex iff its Hessian is positive definite everywhere, and f is convex iff its Hessian is positive semidefinite everywhere.

Suppose that $\mathcal{V}$ is an inner product space with dual space $\mathcal{V}^*$, so that linear functions $\mathcal{V} \to \mathbb{R}$ can be written as $\boldsymbol{u}^T\boldsymbol{x}$ with $\boldsymbol{u}^T \in \mathcal{V}^*$. A half space is defined as $\{(\boldsymbol{x}, y) \,|\, \boldsymbol{u}^T\boldsymbol{x} - \mu \leq y\}$. If f is convex, then the convex set $\mathrm{epi}(f)$ can be represented as intersection of all half spaces which contain it (this is a property of convex sets). But a half space contains $\mathrm{epi}(f)$ iff

$$\mu \geq \sup_{\boldsymbol{x} \in \mathcal{C}} \boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x}),$$

and in fact to represent $\mathrm{epi}(f)$ and therefore f, we only need to consider the halfspaces $(\boldsymbol{u}, \mu)$ with

$$\mu = f^*(\boldsymbol{u}) = \sup_{\boldsymbol{x} \in \mathcal{C}} \boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x}), \tag{A.3}$$

where the domain of f^* is the set of all $\boldsymbol{u}$ such that the supremum exists. f^* is called the *Legendre dual* of f. Intuitively, if f is nicely behaved, then the dual represents f in terms of all tangent planes to f, and f^* maps gradients to offsets. For any $\boldsymbol{u} \in \mathcal{C}^* = \mathrm{dom}(f^*)$, $\boldsymbol{u}^T\boldsymbol{x} - f^*(\boldsymbol{u})$ is a tight affine global lower bound to $f(\boldsymbol{x})$. One might expect that

$$f(\boldsymbol{x}) = \sup_{\boldsymbol{u} \in \mathcal{C}^*} \boldsymbol{u}^T\boldsymbol{x} - f^*(\boldsymbol{u}), \tag{A.4}$$

which is indeed the case, furthermore f^* is convex itself. We have the bound

$$\boldsymbol{u}^T\boldsymbol{x} \leq f(\boldsymbol{x}) + f^*(\boldsymbol{u}), \quad \boldsymbol{x} \in \mathcal{C},\, \boldsymbol{u} \in \mathcal{C}^*. \tag{A.5}$$

The relationship $f \leftrightarrow f^*$ is called *Legendre* or *convex duality*. If $\mathcal{V} = \mathbb{R}^d$, $\mathcal{C}$ is open and f is strictly convex and continuously differentiable, we can solve (A.3) by differentiating: $\boldsymbol{u} = \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$. Due to the strict convexity, this equation can have at most one solution. Thus, $\mathcal{C}^*$ is the gradient space of f, and $\boldsymbol{u} = \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ defines a one-to-one mapping $\boldsymbol{x} \leftrightarrow \boldsymbol{u}$ between *Legendre pairs*. The mapping is inverted via $\boldsymbol{x} = \nabla_{\boldsymbol{u}} f^*(\boldsymbol{u})$, and (A.5) is an equality iff $(\boldsymbol{x}, \boldsymbol{u})$ is a Legendre pair. The duality between $f(\boldsymbol{x})$ and $f^*(\boldsymbol{u})$ is illustrated in Figure A.1.

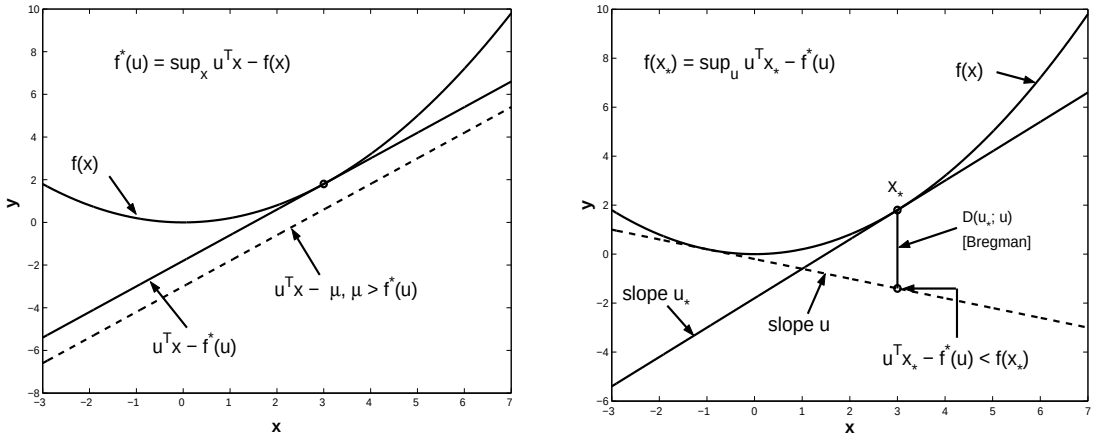

Figure A.1: Illustrations of convex (Legendre) duality. Left: f^* parameterises affine tight lower bounds to f. Right: reconstruction of f from lower bounds given by f^*; Bregman divergence.

For every $(\hat{\boldsymbol{x}}, \boldsymbol{u})$, we have $f(\hat{\boldsymbol{x}}) \geq \boldsymbol{u}^T\hat{\boldsymbol{x}} - f^*(\boldsymbol{u})$, with equality iff $\boldsymbol{u} = \hat{\boldsymbol{u}}$, $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ a Legendre pair. The error (w.r.t. local approximation at $\hat{\boldsymbol{x}}$) made by using the lower bound based on $\boldsymbol{u}$ instead, is quantified by

$$D(\hat{\boldsymbol{u}}; \boldsymbol{u}) = f(\hat{\boldsymbol{x}}) - \boldsymbol{u}^T\hat{\boldsymbol{x}} + f^*(\boldsymbol{u}) = (\hat{\boldsymbol{u}} - \boldsymbol{u})^T\hat{\boldsymbol{x}} + f^*(\boldsymbol{u}) - f^*(\hat{\boldsymbol{u}}),$$

where $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ is a Legendre pair. $D(\hat{\boldsymbol{u}}; \boldsymbol{u})$ is called *Bregman divergence* w.r.t. f^*, and if f^* is strictly convex, then $D(\hat{\boldsymbol{u}}; \boldsymbol{u}) = 0$ iff $\boldsymbol{u} = \hat{\boldsymbol{u}}$. Note that the Bregman divergence is convex in $\boldsymbol{u}$. By applying the same procedure to the dual f, we

obtain a Bregman divergence between elements of $\mathcal{C}$. Maybe the most important Bregman divergence, the relative entropy, is introduced below. See Figure A.1 for a graphical illustration.

Examples for convex dualities are given by the p-norms. If $p, q > 1$ such that $1/p + 1/q = 1$, then $f(\boldsymbol{x}) = (1/p)\|\boldsymbol{x}\|_p^p$ and $f^*(\boldsymbol{u}) = (1/q)\|\boldsymbol{u}\|_q^q$ are duals. Note that $f(\boldsymbol{x}) = (1/2)\boldsymbol{x}^T\boldsymbol{x}$ is self-dual. Another important duality, featuring the relative entropy, is given below.

A useful inequality can be directly deduced from Legendre duality:

Lemma A.4 (Jensen's Inequality) *If f is convex and $\boldsymbol{x}$ is a random variable with finite mean, then*

$$\mathrm{E}\left[f(\boldsymbol{x})\right] \geq f\left(\mathrm{E}[\boldsymbol{x}]\right).$$

If f is strictly convex, then equality here implies $\boldsymbol{x} = \mathrm{E}[\boldsymbol{x}]$ almost surely.

Namely, let $\bar{\boldsymbol{x}} = \mathrm{E}[\boldsymbol{x}]$. Using (A.4), for every $\varepsilon > 0$ we have a $\boldsymbol{u}$ such that $f(\boldsymbol{x}) - \boldsymbol{u}^T(\boldsymbol{x} - \bar{\boldsymbol{x}}) \geq \boldsymbol{u}^T\bar{\boldsymbol{x}} - f^*(\boldsymbol{u}) > f(\bar{\boldsymbol{x}}) + \varepsilon$. Now, take expectations and let $\varepsilon \to 0$.

In the remainder of this section, we introduce some information-theoretic functions whose convexity properties we use extensively in this thesis. For an excellent introduction to information theory, see [34].

Definition A.4 (Relative Entropy) *Let P, Q be two probability measures on the same space with $Q \ll P$, such that the density dQ/dP exists almost everywhere. The* relative entropy *is defined as*

$$\mathrm{D}[Q \,\|\, P] = \mathrm{E}_Q\left[\log \frac{dQ}{dP}\right] = \int \left(\log \frac{dQ}{dP}\right) dQ.$$

If Q is not absolutely continuous w.r.t. P, we set $\mathrm{D}[Q \,\|\, P] = \infty$. It is always non-negative, and equal to 0 iff $Q = P$. The function $(Q, P) \mapsto \mathrm{D}[Q \,\|\, P]$ is strictly convex.

The non-negativity of $\mathrm{D}[Q \,\|\, P]$ follows directly from the strict convexity of $-\log$ and Jensen's Inequality A.4. It is sometimes called *information inequality*.

The convexity of $\mathrm{D}[Q \,\|\, P]$ follows from the log sum inequality (see [34], Theorem 2.7.1). Note that $\mathrm{D}[Q \,\|\, P]$ is not a metric, since it is not symmetric in general and also does not satisfy the triangle inequality in its usual form. An intuitive motivation for $\mathrm{D}[Q \,\|\, P]$ is in terms of coding cost. If we were to encode symbols drawn i.i.d. from Q, but used a code optimal for P, we would lose on average $\mathrm{D}[Q \,\|\, P]$ "nats" per symbol as compared to using an optimal code for Q. Thus, Q may be viewed as "true distribution" and P as approximation to Q, although in some applications these roles are flipped.

Let Q, P be distributions of $\boldsymbol{w}$, absolutely continuous w.r.t. some dominating positive measure. If the mass of $\boldsymbol{w}$ is concentrated on a finite set of size L, say $\{1, \ldots, L\}$, i.e. the dominating measure is the counting measure putting weight 1 on each of $l \in \{1, \ldots, L\}$, then $Q(\boldsymbol{w})$ and $P(\boldsymbol{w})$ are finite distributions and

$$\mathrm{D}[Q \,\|\, P] = \sum_{l=1}^{L} \Pr_Q\{\boldsymbol{w} = l\} \log \frac{\Pr_Q\{\boldsymbol{w} = l\}}{\Pr_P\{\boldsymbol{w} = l\}}. \tag{A.6}$$

If $\boldsymbol{w} \in \mathbb{R}^m$, the dominating measure is typically the Lebesgue measure $d\boldsymbol{w}$ and

$$\mathrm{D}[Q \,\|\, P] = \mathrm{E}_{\boldsymbol{w} \sim Q}\left[\log \frac{Q(\boldsymbol{w})}{P(\boldsymbol{w})}\right]$$

(recall that we use the same notation for a distribution and its density w.r.t. $d\boldsymbol{w}$, i.e. $Q(\boldsymbol{w}) = dQ/d\boldsymbol{w}$). A special case of (A.6) for Bernoulli distributions (skew coins) with probabilities of heads q, p is frequently used in this thesis:

$$\mathrm{D}_{\mathrm{Ber}}[q \,\|\, p] = q \log \frac{q}{p} + (1 - q) \log \frac{1 - q}{1 - p}. \tag{A.7}$$

$\mathrm{D}_{\mathrm{Ber}}$ is convex in (q, p), furthermore $p \mapsto \mathrm{D}_{\mathrm{Ber}}[q \,\|\, p]$ is strictly monotonically increasing for $p \geq q$, mapping $[q, 1)$ to $[0, \infty)$, and strictly decreasing for $p \leq q$, mapping $(0, q]$ to $[0, \infty)$.

$Q \mapsto \mathrm{D}[Q \,\|\, P]$ is convex, and its representation w.r.t. its Legendre dual is given by

$$\mathrm{D}[Q \,\|\, P] = \max_{\lambda} \left(\mathrm{E}_{\boldsymbol{w} \sim Q}\left[\lambda(\boldsymbol{w})\right] - \log \mathrm{E}_{\boldsymbol{w} \sim P}\left[e^{\lambda(\boldsymbol{w})}\right]\right), \tag{A.8}$$

where the maximum is over all $\lambda(\boldsymbol{w})$ measurable w.r.t. $P(\boldsymbol{w})$ (λ is only defined up to an additive constant). Thus, its Legendre dual is the log partition function

$f^*(\lambda) = \log \mathrm{E}_P[\exp(\lambda(\boldsymbol{w}))]$. For Q with $\mathrm{D}[Q \,\|\, P] < \infty$, the dual parameter is $\lambda(\boldsymbol{w}) = \log dQ(\boldsymbol{w})/dP(\boldsymbol{w}) + c$. This is shown in Section 3.2.2, within the proof of Theorem 3.2. The relative entropy also emerges as Bregman divergence. If $\hat{\lambda} = \log d\hat{Q}/dP$ is the choice for the Legendre pair $(\hat{Q}, \hat{\lambda})$, corresponding to $c = 0$, then it is easy to see that $D(\hat{Q}; Q) = \mathrm{D}[Q \,\|\, \hat{Q}]$ (including the case $\infty = \infty$) as Bregman divergence w.r.t. $\mathrm{D}[\cdot \,\|\, P]$.

The relative entropy can be used to derive other information-theoretic functions. If we fix a base measure P_0 (finite, need not be a probability), the *entropy* can be defined as $\mathrm{H}[Q] = -\mathrm{D}[Q \,\|\, P_0]$. In the case of finite distributions (Equation A.6), one normally uses the counting measure for P_0, therefore

$$\mathrm{H}[Q] = -\sum_{l=1}^{L} \Pr_Q\{\boldsymbol{w} = l\} \log \Pr_Q\{\boldsymbol{w} = l\}. \tag{A.9}$$

In this case, $\mathrm{H}[Q] \leq \log L$. For continuous distributions over $\mathbb{R}^d$, the uniform (Lebesgue) measure is not finite. The usual remedy is to subtract off an infinite part of the entropy which does not depend on the argument Q, ending up with the *differential entropy*

$$\mathrm{H}[Q] = -\int Q(\boldsymbol{w}) \, \log Q(\boldsymbol{w}) \, d\boldsymbol{w}. \tag{A.10}$$

Both entropy and differential entropy are concave functions (being the negative of convex ones). The convex duality representation for the "negentropy" $-\mathrm{H}[Q]$ is given by

$$-\mathrm{H}[Q] = \max_{\lambda} \mathrm{E}_Q[\lambda(\boldsymbol{w})] - \log \int e^{\lambda(\boldsymbol{w})} \, d\boldsymbol{w}, \tag{A.11}$$

which can be obtained from (A.8) by using the Lebesgue measure for P and subtracting off the same infinite part on both sides. The corresponding Bregman divergence is $D(\hat{Q}; Q) = \mathrm{D}[Q \,\|\, \hat{Q}]$, where a Legendre pair satisfies $\hat{Q} \propto \exp(\hat{\lambda})$.

A.4 Exponential Families. Gaussians

In this section, we collect definitions and useful properties of general exponential families of distributions and of the Gaussian family.

A.4.1 Exponential Families

Definition A.5 (Exponential Family) *A set $\mathcal{F}$ of distributions with densities*

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^T\boldsymbol{\varphi}(\boldsymbol{x}) + h(\boldsymbol{x}) - \Phi(\boldsymbol{\theta})\right), \quad \boldsymbol{\theta} \in \Theta,$$
$$\Phi(\boldsymbol{\theta}) = \log \int \exp\left(\boldsymbol{\theta}^T\boldsymbol{\varphi}(\boldsymbol{x}) + h(\boldsymbol{x})\right)\, d\boldsymbol{x}$$

is called an exponential family*. Here, $\boldsymbol{\theta}$ are called* natural parameters*, Θ the* natural parameter space*, $\boldsymbol{\varphi}(\boldsymbol{x})$ the* sufficient statistics*, and $\Phi(\boldsymbol{\theta})$ is the* log partition function*. Furthermore, $\boldsymbol{\eta} = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\varphi}(\boldsymbol{x})]$ are called* moment parameters*.*

One of the important reasons for considering exponential families is that the likelihood function for i.i.d. data from $\mathcal{F}$ is a function of the sample average of the sufficient statistics $\boldsymbol{\varphi}(\boldsymbol{x})$ which has the fixed dimensionality of $\boldsymbol{\theta}$, independent of the sample size. Even if a model does not give rise to posteriors in an exponential family (see Section A.6), members of $\mathcal{F}$ can be used as approximating distributions, since new information can be incorporated without increasing the size of the parametric representation. Many familiar distributions form exponential families, such as Gaussians (see Section A.4.3), multinomials, gammas, etc.

The natural parameter space Θ for $\boldsymbol{\theta}$ is always convex. If there are linear or affine dependencies between the components of $\boldsymbol{\varphi}(\boldsymbol{x})$, then some components in $\boldsymbol{\theta}$ are redundant, and the representation is called *overcomplete*. Otherwise, it is called *minimal*. Note that many useful properties hold only (in general) for minimal representations, which are also most useful in practice, however sometimes notationally clumsy to work with. Our approach here is to state general properties for minimal representations only, however use these properties for special overcomplete representations occasionally. This can be justified by adding linear constraints on $\boldsymbol{\theta}$, which do not destroy the convexity of Θ. In the remainder of this section, we assume that the representation of $\mathcal{F}$ is minimal.

The log partition function $\Phi(\boldsymbol{\theta})$ is closely related to the cumulant generating function of $\boldsymbol{\varphi}(\boldsymbol{x})$, $\boldsymbol{x} \sim P(\boldsymbol{x}|\boldsymbol{\theta})$:

$$\log \mathrm{E}_{\boldsymbol{\theta}}\left[\exp(\boldsymbol{\varepsilon}^T\boldsymbol{\varphi}(\boldsymbol{x}))\right] = \Phi(\boldsymbol{\theta} + \boldsymbol{\varepsilon}) - \Phi(\boldsymbol{\theta})$$

which exists iff $\boldsymbol{\theta} + \boldsymbol{\varepsilon} \in \Theta$. Thus, if $\boldsymbol{\theta}$ is in the interior of Θ, the cumulants of $\boldsymbol{\varphi}(\boldsymbol{x})$ are obtained as derivatives of $\Phi(\boldsymbol{\theta})$, especially $\nabla_{\boldsymbol{\theta}}\Phi = \mathrm{E}_{\boldsymbol{\theta}}[\boldsymbol{\varphi}(\boldsymbol{x})] = \boldsymbol{\eta}$ and $\nabla\nabla_{\boldsymbol{\theta}}\Phi = \mathrm{Var}_{\boldsymbol{\theta}}[\boldsymbol{\varphi}(\boldsymbol{x})]$. Since the representation is minimal, we see that $\Phi(\boldsymbol{\theta})$ is strictly convex, and using Legendre duality (Section A.3), we obtain the following

Lemma A.5 (Natural and Moment Parameters) *If $\mathcal{F}$ is an exponential family with minimal representation, then there is a bijective mapping between the natural parameters $\boldsymbol{\theta}$ and the moment parameters $\boldsymbol{\eta}$. The log partition function $\Phi(\boldsymbol{\theta})$ is strictly convex and has the Legendre dual*

$$\Psi(\boldsymbol{\eta}) = \mathrm{E}_{\boldsymbol{\eta}}\left[\log P(\boldsymbol{x}|\boldsymbol{\eta}) - h(\boldsymbol{x})\right],$$

where $\mathrm{E}_{\boldsymbol{\eta}}[\cdot]$ denotes expectation w.r.t. $P(\boldsymbol{x}|\boldsymbol{\eta}) = P(\boldsymbol{x}|\boldsymbol{\theta}(\boldsymbol{\eta}))$. Conversions between $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ are done as follows:

$$\boldsymbol{\eta}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\Phi, \quad \boldsymbol{\theta}(\boldsymbol{\eta}) = \nabla_{\boldsymbol{\eta}}\Psi.$$

Note that $\boldsymbol{\theta}$ are also called *exponential parameters*, and $\boldsymbol{\eta}$ are also known as *mean parameters*.

If $P(\boldsymbol{x}|\boldsymbol{\theta})$ is in $\mathcal{F}$ and $\boldsymbol{x} = (\boldsymbol{y}^T\ \boldsymbol{z}^T)^T$, then the conditional distribution $P(\boldsymbol{y}|\boldsymbol{z})$ is in an exponential family with sufficient statistics $\boldsymbol{\varphi}'(\boldsymbol{y}) = \boldsymbol{\varphi}(\boldsymbol{x})$ and $h'(\boldsymbol{y}) = h(\boldsymbol{x})$ and has natural parameters $\boldsymbol{\theta}$ within this family. The new family typically has an overcomplete representation even if $\mathcal{F}$ is minimal, so a re-parameterisation is necessary if we want to use the properties of minimally parameterised families.

Note that the class of all exponential family distributions is *not* closed w.r.t. marginalisation. For example, if $P(\boldsymbol{x}|\boldsymbol{\theta})$ is a joint of a continuous Gaussian and a discrete multinomial variable, marginalising over the latter results in a mixture of Gaussians which is in general not in an exponential family. A number of exponential subfamilies such as the (multivariate) Gaussian or multinomial ones *are* however closed under marginalisation.

Lemma A.6 (Product of Exponential Distributions) *If $h \equiv 0$, then a product of densities from $\mathcal{F}$ is an unnormalised member of $\mathcal{F}$:*

$$\prod_{j=1}^{m} P(\boldsymbol{x}|\boldsymbol{\theta}_j) = P\left(\boldsymbol{x}\,\middle|\,\sum_{j=1}^{m}\boldsymbol{\theta}_j\right)\exp\left(\Phi\left(\sum_{j=1}^{m}\boldsymbol{\theta}_j\right) - \sum_{j=1}^{m}\Phi(\boldsymbol{\theta}_j)\right),$$

given that $\sum_j \boldsymbol{\theta}_j$ *lies in* Θ.

Given a positive function $f(\boldsymbol{x})$, we can induce a *tilted exponential family* from $\mathcal{F}$ by essentially adding $\log f(\boldsymbol{x})$ to $h(\boldsymbol{x})$ and recomputing the log partition function.

Definition A.6 (Tilted Exponential Family) *If* $\mathcal{F}$ *is an exponential family with natural parameter* $\boldsymbol{\theta}$ *and* $f(\boldsymbol{x})$ *is a positive function such that*

$$\Phi_f(\boldsymbol{\theta}) = \log \mathrm{E}_{P(\cdot|\boldsymbol{\theta})}\left[f(\boldsymbol{x})\right] + \Phi(\boldsymbol{\theta})$$

exists for every $\boldsymbol{\theta}$*, then the* tilted exponential family $\mathcal{F}_f$ *induced by* $f(\boldsymbol{x})$ *from* $\mathcal{F}$ *contains the distributions*

$$P_f(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^T\boldsymbol{\varphi}(\boldsymbol{x}) + h(\boldsymbol{x}) + \log f(\boldsymbol{x}) - \Phi_f(\boldsymbol{\theta})\right) \propto f(\boldsymbol{x})P(\boldsymbol{x}|\boldsymbol{\theta}).$$

$\mathcal{F}_f$ *has the same natural parameter space* Θ *than* $\mathcal{F}$.

Since $\mathcal{F}_f$ is a proper exponential family, the moment parameter of $P_f(\boldsymbol{x}|\boldsymbol{\theta})$ can be computed as derivatives of $\Phi_f(\boldsymbol{\theta})$, i.e.

$$\mathrm{E}_{P_f(\cdot|\boldsymbol{\theta})}[\boldsymbol{\varphi}(\boldsymbol{x})] = \nabla_{\boldsymbol{\theta}} \log \mathrm{E}_{P(\cdot|\boldsymbol{\theta})}\left[f(\boldsymbol{x})\right] + \boldsymbol{\eta}. \tag{A.12}$$

Note that $\mathcal{F}_f$ is the same set of distributions than $\mathcal{F}$ iff $\log f(\boldsymbol{x}) = \boldsymbol{\varepsilon}^T\boldsymbol{\varphi}(\boldsymbol{x})$ for some vector $\boldsymbol{\varepsilon}$ such that $\boldsymbol{\varepsilon} + \Theta = \{\boldsymbol{\varepsilon} + \boldsymbol{\theta} \mid \boldsymbol{\theta} \in \Theta\} = \Theta$.

If we "update" a distribution from $\mathcal{F}$ by multiplying with a positive factor and renormalising, we will end up in $\mathcal{F}$ iff the update factor has the structure of a ratio of members of $\mathcal{F}$.

Definition A.7 (Unnormalised Exponential Family) *If* $\mathcal{F}$ *is an exponential family with natural parameter* $\boldsymbol{\theta} \in \Theta$*, the set of functions*

$$P^U(\boldsymbol{x}|\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^T\boldsymbol{\varphi}(\boldsymbol{x})\right), \quad \boldsymbol{\theta} = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_2,\ \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta,$$

is referred to as unnormalised exponential family $\mathcal{F}^U$ *associated with* $\mathcal{F}$.

Note that members of $\mathcal{F}^U$ are in general no probability densities, and some of them may not be normalisable at all. If $P(\boldsymbol{x}|\boldsymbol{\theta}) \in \mathcal{F}$, $P^U(\boldsymbol{x}|\tilde{\boldsymbol{\theta}}) \in \mathcal{F}^U$, then $P(\boldsymbol{x}|\boldsymbol{\theta})P^U(\boldsymbol{x}|\tilde{\boldsymbol{\theta}})$ is proportional to a member of $\mathcal{F}$ (namely, to $P(\boldsymbol{x}|\boldsymbol{\theta} + \tilde{\boldsymbol{\theta}})$) iff $\boldsymbol{\theta} + \tilde{\boldsymbol{\theta}} \in \Theta$. Note also that $1 \equiv P^U(\boldsymbol{x}|\mathbf{0}) \in \mathcal{F}^U$.

Lemma A.7 (Relative Entropy) *The* relative entropy

$$\mathrm{D}[P(\boldsymbol{x}|\boldsymbol{\theta}) \,\|\, P(\boldsymbol{x}|\boldsymbol{\theta}^*)] = \mathrm{E}_{\boldsymbol{\theta}} \left[\log P(\boldsymbol{x}|\boldsymbol{\theta}) - \log P(\boldsymbol{x}|\boldsymbol{\theta}^*)\right]$$

(see Definition A.4), also denoted $\mathrm{D}[\boldsymbol{\theta} \,\|\, \boldsymbol{\theta}^*]$ *or* $\mathrm{D}[\boldsymbol{\eta} \,\|\, \boldsymbol{\eta}^*]$*, is given by*

$$\begin{aligned} \mathrm{D}[\boldsymbol{\theta} \,\|\, \boldsymbol{\theta}^*] &= \boldsymbol{\eta}^T(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \Phi(\boldsymbol{\theta}^*) - \Phi(\boldsymbol{\theta}) \\ &= (\boldsymbol{\theta}^*)^T(\boldsymbol{\eta}^* - \boldsymbol{\eta}) + \Psi(\boldsymbol{\eta}) - \Psi(\boldsymbol{\eta}^*). \end{aligned} \tag{A.13}$$

From (A.13), we see that $\mathrm{D}[\boldsymbol{\theta} \,\|\, \boldsymbol{\theta}^*]$ is convex in $\boldsymbol{\theta}^*$, and $\mathrm{D}[\boldsymbol{\eta} \,\|\, \boldsymbol{\eta}^*]$ is convex in $\boldsymbol{\eta}$.

A.4.2 I-Projections

The relative entropy (Definition A.4) is a natural replacement for Euclidean distance (or its square) when dealing with manifolds of probability distributions, allowing for generalisations of important geometrical concepts such as orthogonal projection, triangle inequalities and Pythagorean theorems. The equivalent of orthogonal projection has been introduced by Csiszár [43] as *I-projections*.

Definition A.8 (I-projections) *Let* $\mathcal{P}$ *be some closed convex set (see Section A.3) of distributions* P*, and let* Q *be some distribution. The* e-projection *of* Q *onto* $\mathcal{P}$ *is*

$$P^{[e]} = \operatorname*{argmin}_{P \in \mathcal{P}} \mathrm{D}[P \,\|\, Q],$$

and the m-projection *of* Q *onto* $\mathcal{P}$ *is*

$$P^{[m]} = \operatorname*{argmin}_{P \in \mathcal{P}} \mathrm{D}[Q \,\|\, P].$$

They are both examples of I-projections*.*

Note that both I-projections are well-defined, since $\mathcal{P}$ is convex and the relative entropy is convex in both arguments. Also, they project Q onto itself iff $Q \in \mathcal{P}$. Suppose that $\mathcal{P}$ is an exponential family (Definition A.5). Then, $\mathrm{D}[Q \,\|\, P] = \Phi(\boldsymbol{\theta}) - \boldsymbol{\theta}^T \mathrm{E}_Q[\boldsymbol{\varphi}(\boldsymbol{x})]$, and $P^{[m]}$ is given by the moment parameter $\boldsymbol{\mu}^{[m]} = \mathrm{E}_Q[\boldsymbol{\varphi}(\boldsymbol{x})]$. In this important special case, the *m-projection* is equivalent to *moment matching* between Q and $P^{[m]}$. Another example is given in Section A.6.4.

A.4.3 Gaussian Variables

Definition A.9 (Gaussian Distribution) *A variable $\boldsymbol{x} \in \mathbb{R}^d$ has a (non-degenerate) Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ iff its p.d.f. is given by*

$$P(\boldsymbol{x}) = N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = |2\pi\boldsymbol{\Sigma}|^{-d/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right).$$

A less stringent definition is the following: $\boldsymbol{x}$ has Gaussian distribution if for every vector $\boldsymbol{a} \in \mathbb{R}^d$, the variable $\boldsymbol{a}^T\boldsymbol{x}$ has either a non-degenerate Gaussian distribution or is (almost surely) constant. This allows for degenerate distributions with supports confined to affine subspaces of $\mathbb{R}^d$.

The Gaussian family is the most important family of continuous distributions, both in theory and in practice, for a number of reasons. It has superb closeness properties, e.g. under linear transformations, conditioning and marginalisation. The Gaussian arises as the ideal noise or error distribution from the central limit theorem. Among all distributions with a given mean and covariance matrix, it has the least structure, i.e. maximises the differential entropy. All cumulants of order higher than 2 vanish (the cumulant generating function is $\boldsymbol{\theta}^T\boldsymbol{\mu} + (1/2)\boldsymbol{\theta}^T\boldsymbol{\Sigma}\boldsymbol{\theta}$, a quadratic). Gaussian variables which are not correlated, are independent.

In this thesis, Gaussians play a large role. Gaussian processes (GPs, see Section 2.1) induce Gaussian distributions when evaluated on finite point sets. In the remainder of this section, we introduce some notation which will make it easier to manipulate Gaussian expressions. In Section A.4.3, we collect a number of Gaussian formulae which are used in this thesis.

The Gaussian family is an exponential family, and from Definition A.7 we have

Definition A.10 (Unnormalised Gaussian)

$$N^U(\boldsymbol{x}|\boldsymbol{r}, \boldsymbol{V}, c) = \exp\left(-\frac{1}{2}\boldsymbol{x}^T\boldsymbol{V}\boldsymbol{x} + \boldsymbol{r}^T\boldsymbol{x} - \frac{1}{2}c\right),$$

where $\boldsymbol{V}$ is symmetric (w.l.o.g.), but need not be positive definite. Furthermore, $N^U(\boldsymbol{x}|\boldsymbol{r}, \boldsymbol{V}) = N^U(\boldsymbol{x}|\boldsymbol{r}, \boldsymbol{V}, 0)$. A centred version is

$$N^{UC}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{V}) = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{\mu})\right).$$

Note that for a given N^U, there may exist none, one or many N^{UC} which are proportional. $N^U(\boldsymbol{x}|\boldsymbol{r},\boldsymbol{V})$ is proportional to a Gaussian density iff $\boldsymbol{V}$ is positive definite. We can use $\boldsymbol{r}$ and $\boldsymbol{V}$ as natural parameters $\boldsymbol{\theta}$, together with the sufficient statistics $\boldsymbol{x}$ and $-(1/2)\boldsymbol{x}\boldsymbol{x}^T$. This parameterisation is not minimal, due to the symmetry constraint on $\boldsymbol{V}$, but is easier to work with than a minimal one. The corresponding moment parameters are $\boldsymbol{\mu} = \boldsymbol{V}^{-1}\boldsymbol{r}$ and $\boldsymbol{\Sigma} = \boldsymbol{V}^{-1}$, i.e. mean and covariance matrix[6] of the Gaussian $P(\boldsymbol{x}|\boldsymbol{\theta}) \propto N^U(\boldsymbol{x}|\boldsymbol{r},\boldsymbol{V})$. The conversion between natural and moment parameters requires the inversion of a positive definite matrix. The log partition function is given by

$$\Phi(\boldsymbol{\theta}) = \frac{1}{2}\left(\boldsymbol{r}^T\boldsymbol{V}^{-1}\boldsymbol{r} - \log\left|2\pi\boldsymbol{V}^{-1}\right|\right).$$

The set of all N^U is the associated unnormalised exponential family. Conversion formulae are given in the following

Lemma A.8 (Conversions) *Conversions between (unnormalised) Gaussians:*

- $N(\boldsymbol{\mu},\boldsymbol{\Sigma}) \leftrightarrow N^{UC}(\boldsymbol{\mu},\boldsymbol{V})$

$$N(\boldsymbol{\mu},\boldsymbol{\Sigma}) = |2\pi\boldsymbol{\Sigma}|^{-1/2}\, N^{UC}\left(\boldsymbol{\mu},\boldsymbol{\Sigma}^{-1}\right),$$
$$N^{UC}(\boldsymbol{\mu},\boldsymbol{V}) = \left|2\pi\boldsymbol{V}^{-1}\right|^{1/2} N\left(\boldsymbol{\mu},\boldsymbol{V}^{-1}\right), \quad \boldsymbol{V} \text{ pos. def.}$$

- $N^{UC}(\boldsymbol{\mu},\boldsymbol{V}) \leftrightarrow N^U(\boldsymbol{r},\boldsymbol{V},c)$

$$N^{UC}(\boldsymbol{\mu},\boldsymbol{V}) = N^U\left(\boldsymbol{V}\boldsymbol{\mu},\boldsymbol{V},\boldsymbol{\mu}^T\boldsymbol{V}\boldsymbol{\mu}\right),$$
$$N^U(\boldsymbol{r},\boldsymbol{V},c) = N^{UC}(\boldsymbol{\mu},\boldsymbol{V})\exp\left(-\frac{1}{2}\left(c - \boldsymbol{\mu}^T\boldsymbol{r}\right)\right), \quad \boldsymbol{V}\boldsymbol{\mu} = \boldsymbol{r}.$$

As specialisation of Definition A.6, for some positive function $f(\boldsymbol{x})$, let

$$N_f^U(\boldsymbol{x}|\boldsymbol{r},\boldsymbol{V},c) = f(\boldsymbol{x})N^U(\boldsymbol{x}|\boldsymbol{r},\boldsymbol{V},c).$$

If N_f^U is normalisable for all positive definite $\boldsymbol{V}$, then $\hat{P} \propto N_f^U$ form a tilted exponential family, and we can compute the moments of $\hat{P}$ using derivatives of

$$\log Z(\boldsymbol{V},\boldsymbol{r}) = \log \mathrm{E}_{(\boldsymbol{V},\boldsymbol{r})}\left[f(\boldsymbol{x})\right],$$

[6]Strictly speaking, the second moment parameter is $\mathrm{E}_{\boldsymbol{\theta}}[-(1/2)\boldsymbol{x}\boldsymbol{x}^T] = -(1/2)(\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T)$.

namely

$$\mathrm{E}_{\hat{P}}[\boldsymbol{x}] = \nabla_{\boldsymbol{r}} \log Z + \boldsymbol{\mu}, \quad \mathrm{Var}_{\hat{P}}[\boldsymbol{x}] = \nabla\nabla_{\boldsymbol{r}} \log Z + \boldsymbol{\Sigma} \tag{A.14}$$

(see Section A.4.1).

Manipulating Gaussians in the sense of forming linear combinations, marginalisation and conditioning boils down to simple linear algebra, as is summarised in the following lemmas.

Lemma A.9 (Conditional Gaussian) *Let $\boldsymbol{x} \in \mathbb{R}^n$ be distributed as $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. If $I \subset \{1, \ldots, n\}$, $R = \{1, \ldots, n\} \setminus I$, then $\boldsymbol{x}_R|\boldsymbol{x}_I$ is Gaussian with covariance matrix*

$$\boldsymbol{\Sigma}^{(R|I)} = \left(\boldsymbol{\Sigma}^{-1}\right)_R^{-1} = \boldsymbol{\Sigma}/\boldsymbol{\Sigma}_I = \boldsymbol{\Sigma}_R - \boldsymbol{\Sigma}_{R,I} \left(\boldsymbol{\Sigma}_I\right)^{-1} \boldsymbol{\Sigma}_{I,R}$$

and mean

$$\boldsymbol{\mu}^{(R|I)} = \boldsymbol{\mu}_R + (\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_R)(\boldsymbol{\Sigma}_{R,I}/\boldsymbol{\Sigma})(\boldsymbol{x}_I - \boldsymbol{\mu}_I) = \boldsymbol{\mu}_R + \boldsymbol{\Sigma}_{R,I}(\boldsymbol{\Sigma}_I)^{-1}(\boldsymbol{x}_I - \boldsymbol{\mu}_I).$$

This can be shown by taking $N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ as function of $\boldsymbol{x}_R$ and use Lemma A.1. Note that the covariance matrix of $\boldsymbol{x}_R|\boldsymbol{x}_I$ does not depend on $\boldsymbol{x}_I$.

Lemma A.10 (Sums and Affine Transforms) *Let $\boldsymbol{x}_i \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ be independent Gaussian variables and $\boldsymbol{y} = \sum_i \boldsymbol{A}_i \boldsymbol{x}_i + \boldsymbol{b}$. Then,*

$$\boldsymbol{y} \sim N\left(\boldsymbol{y} \,\middle|\, \sum_i \boldsymbol{A}_i \boldsymbol{\mu}_i + \boldsymbol{b}, \sum_i \boldsymbol{A}_i \boldsymbol{\Sigma}_i \boldsymbol{A}_i^T\right).$$

Marginalisation is a special case: if $\boldsymbol{x} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $I \subset \{1, \ldots, d\}$ does not contain duplicates, then $\boldsymbol{x}_I \sim N(\boldsymbol{\mu}_I, \boldsymbol{\Sigma}_I)$.

For unnormalised Gaussians, we note

$$N^U\left(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \,\middle|\, \boldsymbol{r}, \boldsymbol{V}, c\right) = N^U\left(\boldsymbol{x} \,\middle|\, \boldsymbol{A}^T(\boldsymbol{V}\boldsymbol{b} + \boldsymbol{r}), \boldsymbol{A}^T\boldsymbol{V}\boldsymbol{A}, c + \boldsymbol{b}^T\boldsymbol{V}\boldsymbol{b}\right).$$

Products of unnormalised Gaussians are unnormalised Gaussians again, and the natural parameters combine additively (see Lemma A.6).

Lemma A.11 (Products) *Let there be m unnormalised Gaussians over $\boldsymbol{x} \in \mathbb{R}^n$, each of them either of the form $N^{UC}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{V}_i)$ or $N^{U}(\boldsymbol{x}|\boldsymbol{r}_i, \boldsymbol{V}_i, c_i)$. Convert the former via*

$$\boldsymbol{r}_i = \boldsymbol{V}_i\boldsymbol{\mu}_i, \quad c_i = \boldsymbol{\mu}_i^T\boldsymbol{V}_i\boldsymbol{\mu}_i.$$

Define $\boldsymbol{V}$, $\boldsymbol{r}$ and $\boldsymbol{\mu}$ as

$$\boldsymbol{V} = \sum_i \boldsymbol{V}_i, \quad \boldsymbol{r} = \sum_i \boldsymbol{r}_i, \quad \boldsymbol{\mu} = \boldsymbol{V}^{-1}\boldsymbol{r},$$

where we assume that $\boldsymbol{V}$ is invertible.[7] *Then,*

$$\begin{aligned}\prod_{i=1}^{m} N^{U}(\boldsymbol{x}|\boldsymbol{r}_i, \boldsymbol{V}_i, c_i) &= N^{U}(\boldsymbol{x}|\boldsymbol{r}, \boldsymbol{V}, \sum_i c_i) \\ &= N^{UC}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{V}) \times \exp\left(\frac{1}{2}\boldsymbol{r}^T\boldsymbol{\mu} - \frac{1}{2}\sum_{i=1}^{m} c_i\right).\end{aligned} \tag{A.15}$$

If $\boldsymbol{V}$ is positive definite, the first factor on the right is simply $|2\pi\boldsymbol{\Sigma}|^{1/2}N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} = \boldsymbol{V}^{-1}$. Under additional conditions, we can obtain a recursive formula which we state for $m = 2$ (for larger m, use Lemma A.11 to combine $m - 1$ factors).

Lemma A.12 (Products, Recursive Formulation) *Under the conditions of Lemma A.11 ($m = 2$) we assume that $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{V}_1^{-1} + \boldsymbol{V}_2^{-1}$ are invertible. Then,*

$$N^{UC}(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{V}_1)N^{UC}(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{V}_2) = N^{UC}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{V})N^{UC}\left(\boldsymbol{\mu}_1\middle|\boldsymbol{\mu}_2, \left(\boldsymbol{V}_1^{-1} + \boldsymbol{V}_2^{-1}\right)^{-1}\right). \tag{A.16}$$

If furthermore both $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$ are positive definite and $\boldsymbol{\Sigma}_i = \boldsymbol{V}_i^{-1}$, this becomes the familiar equation

$$N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})N(\boldsymbol{\mu}_1|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2).$$

Lemma A.13 (Differential and Relative Entropy) *The differential entropy of a Gaussian is*

$$\mathrm{H}[N(\boldsymbol{\mu}, \boldsymbol{\Sigma})] = \frac{1}{2}\log\left|2\pi\, e\, \boldsymbol{\Sigma}\right|.$$

[7] The single $\boldsymbol{V}_i$ need not be invertible.

The relative entropy (see Definition A.7) between two Gaussians is given by

$$\begin{aligned}\mathrm{D}[N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \,\|\, N(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)] = \frac{1}{2} \log \left|\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right| + \frac{1}{2} \operatorname{tr}\left(\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right)^{-1} - \boldsymbol{I}\right) \\ + \frac{1}{2} \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right)^T \boldsymbol{\Sigma}_0^{-1} \left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\right).\end{aligned} \tag{A.17}$$

A.5 Pattern Recognition

In this section, we briefly introduce the general problem this thesis is mainly concerned with. In the *classification* or *pattern recognition problem*, we are given data $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, n\},\ \boldsymbol{x}_i \in \mathcal{X},\ y_i \in \mathcal{Y}$, sampled *independently and identically distributed (i.i.d.)* from an unknown *data distribution* over $\mathcal{X} \times \mathcal{Y}$. Here, $\mathcal{Y}$ is a finite set. If $C = |\mathcal{Y}| > 2$, we will assume that $\mathcal{Y} = \{1, \ldots, C\}$, while for $C = 2$, $\mathcal{Y} = \{-1, +1\}$. The latter situation is called the *binary classification problem*, while the case $C > 2$ is referred to as *multi-class problem*. The sample S is also called *training set*. Our goal is to compute a *classification function* or *classifier* $c : \mathcal{X} \to \mathcal{Y}$ from S which has small *generalisation error*

$$\operatorname{gen}(c) = \Pr\left\{c(\boldsymbol{x}_*) \neq y_*\right\} = \mathrm{E}\left[\mathrm{I}_{\{c(\boldsymbol{x}_*) \neq y_*\}}\right],$$

where $(\boldsymbol{x}_*, y_*)$ is sampled from the data distribution, independently of S. c has to be computed from S only, without any further definite knowledge about the data distribution. This process is called *training*. A more ambitious goal is to estimate the conditional data distribution $P(y_*|\boldsymbol{x}_*)$ or related moments thereof directly, leading to so-called *predictors*. In the context of binary classification, many classifiers are defined in terms of *discriminants*, i.e. real-valued functions f for which we obtain classifiers by thresholding: $c(\boldsymbol{x}) = \operatorname{sgn}(f(\boldsymbol{x}) - \theta)$, where θ is a threshold. This notion can be generalised to the multi-class problem by using one discriminant f_y per class and setting $c(\boldsymbol{x}) = \operatorname{argmax}_y f_y(\boldsymbol{x})$.

Functions of the training sample S are referred to as *(empirical) statistics*. An important statistic is the *empirical* or *training error*

$$\operatorname{emp}(c, S) = \frac{1}{n} \sum_{i=1}^{n} \mathrm{I}_{\{c(\boldsymbol{x}_i) \neq y_i\}}.$$

Note that both generalisation and empirical error are defined as expectations of the so-called *zero-one loss* $l_{0/1}(c) = \mathrm{I}_{\{c(\boldsymbol{x})\neq y\}}$. This scenario can be generalised to other non-negative *loss functions* of predictors, in which case the expected loss over the data distribution is called *(true) risk* and the expected loss over the *empirical distribution* $n^{-1}\sum_{i=1}^{n}\delta_{(\boldsymbol{x}_i,y_i)}$ is called *empirical risk*. This framework encompasses other statistical problems such as *regression estimation* where $\mathcal{Y} = \mathbb{R}$, and our task is to estimate $\mathrm{E}[y|\boldsymbol{x}]$ as a function of $\boldsymbol{x}$.

For any fixed classifier c, the empirical error will converge to the generalisation error almost surely, by the strong law of large numbers. But if c is chosen depending on S, e.g. to minimise the empirical error over some broad set of candidates, we will typically not end up with the best candidate w.r.t. $\mathrm{gen}(\cdot)$, in fact not even with an acceptable performance. This problem is known as the *overfitting effect* and can be attributed to the fact that trying to fit the training data overly well leads to our choice being too dependent on the noise in S. One way around this is to penalise supposedly overcomplex candidates in the choice of c, this is called *regularisation*. A more principled, yet often also more costly solution is *Bayesian analysis* (see Section A.6) where a classifier is constructed as expectation over a set of candidates or models. Also here, overcomplex candidates are usually given low weights in the expectation. One of the key points we would like to make in this thesis is that the notion of *complexity* cannot be defined in a both universal[8] and practically feasible way. It has to depend on the context of the statistical problem and on available prior knowledge about characteristics and constraints.

A.6 Bayesian Inference and Approximations

In this section, we introduce some aspects of Bayesian analysis relevant to this thesis. For more detailed accounts, see [15, 14, 23]. We also briefly sketch the basics of some general approximation techniques for Bayesian inference which are relevant to machine learning. An excellent up-to-date introduction to many of these is given in [84].

[8] A universal definition would be *Kolmogorov complexity* (e.g., [34], Chap. 7) which is however not practically feasible to use.

A.6.1 Probabilistic Modelling

A probabilistic model is a simplified abstract description of a (partially) observable data source. Consider the binary classification problem defined in Section A.5. The *observed variables* of the domain are the input points $\boldsymbol{x} \in \mathcal{X}$ and the targets (or labels) $y \in \{-1, +1\}$. A simple model is obtained by introducing a *latent variable* $u \in \mathbb{R}$ together with the assumption that $y \perp \boldsymbol{x} \mid u$. The introduction of u can be motivated by a variety of reasons. First, our prior assumptions include the notion of *smoothness*: the targets at two close-by input points should *a priori* be the same with high probability. To express this, we need to include the probability $P(y|\boldsymbol{x})$ amongst the variables, although it cannot be observed. Second, it is easier to model $P(u|\boldsymbol{x})$ and $P(y|u)$ independently than $P(y|\boldsymbol{x})$ directly. The model description is completed by choosing *prior distributions* for $P(u|\boldsymbol{x})$ and $P(y|u)$.

The specification of $P(u|\boldsymbol{x})$ is sometimes called the *systematic component* of the model, while the specification of $P(y|u)$ is the *random component*. The former can be specified in a *parametric* or a *non-parametric* way: postulate a parametric family $P(u|\boldsymbol{x}, \boldsymbol{w}) = \delta_{u(\boldsymbol{x};\boldsymbol{w})}(u)$ and a prior $P(\boldsymbol{w})$ with the dimensionality of $\boldsymbol{w}$ independent of dataset sizes, or place a prior distribution directly on the probabilistic relationship $\boldsymbol{x} \mapsto u$ which penalises $u(\boldsymbol{x})$ by low probability assignment for violating prior assumptions, e.g. for behaving non-smooth, being discontinuous, etc.[9] Non-parametric modelling and relationships to parametric models are discussed in some detail in Section 2.1. *Semi-parametric* models combine both ways by for example modelling $u(\boldsymbol{x})$ as sum of a parametric and a non-parametric "residual" part. Semi-parametric models are useful to test a parametric modelling assumption by observing the relevance of the non-parametric residual part on the prediction of u.

Noise distributions $P(y|u)$ for several standard models are given in Section 2.1.2. Both $P(y|u)$ and $P(u|\boldsymbol{x})$ can depend on further parameters, called *hyperparameters*, their prior distributions are called *hyperpriors*. Note that for a parametric

[9] Some care is required here, because $u(\boldsymbol{x})$ in general does not have to be a deterministic function, but can be a random process for which characteristics like smoothness and continuity are defined differently, see Section 2.1.1.

model, the *primary parameter* vector $\boldsymbol{w}$ is not treated as hyperparameter. Note that we have not said anything about the distribution of the input points $\boldsymbol{x}$. These are examples of *covariates*, variables which can be considered given at any time when a prediction of the target variables is required. To sum up, our domain partitions into observed and latent variables, the former into observed target variables and covariates, the latter into latent target variables and so-called *nuisance variables*. We are interested in predicting the target variables from covariates and a training sample S. In our example, y is the only target variable, while u is a nuisance variable. The main assumption behind this model is that the different cases $(\boldsymbol{x}_i, y_i)$ are sampled i.i.d., given a common latent parameter $\boldsymbol{w}$ (or latent process $u(\cdot)$) sampled from the prior, *i.e.* that the data distribution satisfies *exchangeability* (de Finetti's theorem, see [15]).

A.6.2 Bayesian Analysis

Bayesian inference and prediction, as central part of Bayesian analysis work as follows. Given a complete model for all variables except the covariates, we condition on the observed data S and marginalise over the nuisance (latent) variables to obtain the posterior distribution

$$P(\boldsymbol{\theta}|S) \propto P(\boldsymbol{\theta})P(S|\boldsymbol{\theta}) = P(\boldsymbol{\theta}) \int P(S, \boldsymbol{h}|\boldsymbol{\theta})\, d\boldsymbol{h}.$$

Here, $\boldsymbol{h}$ are the nuisance variables, $P(S, \boldsymbol{h}|\boldsymbol{\theta})$ is called *complete data likelihood*, and $P(S|\boldsymbol{\theta})$ is called *observed data likelihood*. In our example, $\boldsymbol{h} = \boldsymbol{u}$. By the exchangeability assumption,

$$P(S|\boldsymbol{\theta}) = \prod_{i=1}^{n} P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}), \quad P(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \int P(y_i|u_i)P(u_i|\boldsymbol{x}_i, \boldsymbol{\theta})\, du_i.$$

For the same reason, the *predictive distribution* for a test case $(\boldsymbol{x}_*, y_*)$ is given by

$$P(y_*|\boldsymbol{x}_*, S) = \int P(y_*|\boldsymbol{x}_*, \boldsymbol{\theta})P(\boldsymbol{\theta}|S)\, d\boldsymbol{\theta}.$$

Thus, Bayesian inference amounts to updating the *prior belief* $P(\boldsymbol{\theta})$ into the *posterior belief* $P(\boldsymbol{\theta}|S)$. Predictions are then obtained as posterior averages.

The computational requirements of Bayesian inference can be prohibitive, and techniques to approximate Bayesian computations such as posterior expectations are essential in a Bayesian's toolbox. We discuss some of these in Section A.6.3. A frequently used general *empirical Bayesian* method for marginalising over nuisance hyperparameters is *marginal likelihood maximisation* or *maximum likelihood II*. Let $\boldsymbol{\theta}$ be split into primary parameters $\boldsymbol{w}$ and hyperparameters $\boldsymbol{\alpha}$, and $P(\boldsymbol{\theta}) = P(\boldsymbol{w}|\boldsymbol{\alpha})P(\boldsymbol{\alpha})$. Typically, $\boldsymbol{\alpha}$ is of much smaller dimension than $\boldsymbol{w}$. The posterior is $P(\boldsymbol{\theta}|S) = P(\boldsymbol{w}|S, \boldsymbol{\alpha})P(\boldsymbol{\alpha}|S)$. If S is sufficiently large, $P(\boldsymbol{\alpha}|S)$ will usually be highly peaked around its mode $\hat{\boldsymbol{\alpha}}$ due to central limit behaviour, and we can replace it by $\delta_{\hat{\boldsymbol{\alpha}}}(\boldsymbol{\alpha})$. This is an example of a *maximum a posteriori (MAP)* approximation. Now, posterior expectations can be approximated as

$$\mathrm{E}_{P(\boldsymbol{\theta}|S)}\left[f(\boldsymbol{\theta})\right] = \mathrm{E}_{P(\boldsymbol{w}|S,\hat{\boldsymbol{\alpha}})}\left[f(\boldsymbol{w}, \hat{\boldsymbol{\alpha}})\right].$$

$\hat{\boldsymbol{\alpha}}$ is the maximiser of $\log P(S, \boldsymbol{\alpha}) = \log P(S|\boldsymbol{\alpha}) + \log P(\boldsymbol{\alpha})$, where

$$P(S|\boldsymbol{\alpha}) = \int P(S|\boldsymbol{w}, \boldsymbol{\alpha})P(\boldsymbol{w}|\boldsymbol{\alpha})\, d\boldsymbol{w}$$

is called *marginal (data) likelihood*. Since $\log P(\boldsymbol{\alpha})$ is typically a simple function of $\boldsymbol{\alpha}$, the main task is to optimise $\log P(S|\boldsymbol{\alpha})$ w.r.t. $\boldsymbol{\alpha}$. In many situations, the log marginal likelihood cannot be computed tractably either, due to the integral over $\boldsymbol{w}$, and further approximations have to be considered. Finding and plugging in $\hat{\boldsymbol{\alpha}}$ is a special case of (Bayesian) *model selection*.

A.6.3 Approximations to Bayesian Inference

Let $f(\boldsymbol{\theta})$ be a positive function, where for simplicity we assume that $\mathrm{dom}(f) = \mathbb{R}^d$. Let $I = \int f(\boldsymbol{\theta})\, d\boldsymbol{\theta} < \infty$, and $P(\boldsymbol{\theta}) = f(\boldsymbol{\theta})/I$. A general approximation technique should render $\tilde{I} \approx I$, furthermore an approximation $Q(\boldsymbol{\theta})$ to $P(\boldsymbol{\theta})$ such that $\mathrm{E}_P[g(\boldsymbol{\theta})] \approx \mathrm{E}_Q[g(\boldsymbol{\theta})]$ for other functions g, and the latter expectation can be computed feasibly. The focus here is on methods which can work well even if d is large, given that $P(\boldsymbol{\theta})$ is fairly peaked and does not have heavy tails.

The oldest approximation technique for Bayesian inference was introduced by Laplace to approximate the posterior for a binomial model, it is called *Laplace*

or *saddle-point approximation*. Suppose that $-\log f(\boldsymbol{\theta})$ has a unique minimum point $\hat{\boldsymbol{\theta}}$ and is twice continuously differentiable in a open set containing $\hat{\boldsymbol{\theta}}$. Then, the Hessian $\boldsymbol{H}$ is positive definite at $\hat{\boldsymbol{\theta}}$ and we can approximate

$$f(\boldsymbol{\theta}) \approx \exp\left(\log f(\hat{\boldsymbol{\theta}}) - \frac{1}{2}(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^T \boldsymbol{H} (\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})\right).$$

By normalising the approximation, we have $Q(\boldsymbol{\theta}) = N(\hat{\boldsymbol{\theta}}, \boldsymbol{H}^{-1})$ and $\log \tilde{I} = f(\hat{\boldsymbol{\theta}}) - (1/2)\log|2\pi\boldsymbol{H}|$. Note that this approximation is entirely local, depending only on the value and the curvature of f around its mode. In special situations, the Laplace approximation can be applied to multimodal $P(\boldsymbol{\theta})$, namely if the modes are clearly separated and all have the same local shape. On the other hand, it can perform poor even on unimodal distributions due to its local nature, and it is not applicable if f is not twice continuously differentiable around its mode.

Variational approximations work by reformulating the approximation problem into an optimisation problem, typically the maximisation of a lower bound. One approach is to bound $f(\boldsymbol{\theta}) \geq g(\boldsymbol{\theta}, \boldsymbol{\lambda})$ such that expectations w.r.t. $g(\boldsymbol{\theta}, \boldsymbol{\lambda})$ can be done feasibly for every $\boldsymbol{\lambda}$. Then, $I \geq \int g(\boldsymbol{\theta}, \boldsymbol{\lambda})\, d\boldsymbol{\theta}$ for every $\boldsymbol{\lambda}$. The optimisation problem is now to choose $\boldsymbol{\lambda}$ in order to maximise the lower bound on I. For the resulting $\boldsymbol{\lambda}^*$, we have $\tilde{I} = \int g(\boldsymbol{\theta}, \boldsymbol{\lambda}^*)\, d\boldsymbol{\theta}$ and $Q(\boldsymbol{\theta}) = g(\boldsymbol{\theta}, \boldsymbol{\lambda}^*)/\tilde{I}$. A different, more global variational approach can be derived from the convex duality (A.11) between negentropy and log partition function (substitute $\lambda(\boldsymbol{\theta}) = \log f(\boldsymbol{\theta})$), leading to

$$\log I = \log \int e^{\log f(\boldsymbol{\theta})}\, d\boldsymbol{\theta} \geq \mathrm{E}_Q[\log f(\boldsymbol{\theta})] + \mathrm{H}[Q] \tag{A.18}$$

for any distribution $Q(\boldsymbol{\theta})$. The difference between $\log I$ and the lower bound (i.e. the Bregman divergence) is given by $\mathrm{D}[Q \,\|\, P]$ (note that $(\log f, P)$ is a Legendre pair). We can now choose Q amongst a tractable family in order to maximise the lower bound. In contrast to the other variational technique described above, this minimises at the same time the relative entropy $\mathrm{D}[\cdot \,\|\, P]$, bringing us closer to the target distribution P. Note that as opposed to the Laplace approximation, the variational techniques render a guaranteed lower bound on I. This is reassuring if our goal is to maximise I w.r.t. other parameters, but as long as no corresponding

upper bound is available, we have no method of checking the tightness of the lower bound which would be considerably easier than the original approximation task (a general method to upper bound the log partition function for sparsely connected graphical models is given in [204], and a method for two-layer networks with binary variables can be found in [80]). Moreover, if our goal is to approximate I or $P(\boldsymbol{\theta})$, insisting on a lower bound seems fairly restrictive. A variant of the second variational method, called *variational Bayes* has recently received a lot of attention [7, 62, 19]. Variational Bayes is essentially a special case of the *variational mean field* approximation, in which the family to draw Q from is defined via factorisation constraints. This allows for a simple analytic update of one of the factors, given all the others are fixed, and this process can be iterated. Note that historically methods are called "mean field" approximations if they use completely factorised variational distributions, and the obvious extension to partial factorisations is called "generalised mean field" or "structural mean field", but we do not follow this nomenclature. A general framework for variational Bayes has been given in [62], and general user-friendly software is available [19].

Given that variational mean field approximations can lead to rather poor results, a range of improvements have been suggested. A general ansatz is to look for approximations to a so-called negative free energy $I = \log \int \exp(g(\boldsymbol{\theta}))\, d\boldsymbol{\theta}$. From Section A.3 we know that the convex dual is the negentropy, so that

$$\log \int \exp(g(\boldsymbol{\theta}))\, d\boldsymbol{\theta} = \max_Q \mathrm{E}_Q[g(\boldsymbol{\theta})] + \mathrm{H}[Q],$$

and the optimal Q (such that (g, Q) is a Legendre pair) is $Q(\boldsymbol{\theta}) \propto \exp(g(\boldsymbol{\theta}))$, the desired target distribution. The mean field approximation maximises the lower bound w.r.t. factorised Q distributions, and if our goal is to approximate some marginals of the target distribution, we may use the corresponding marginals of the maximiser Q. A different approach is to approximate the r.h.s. (in particular the intractable entropy term) by a function of certain marginals of Q only, disregarding the fact that many sets of marginals we deal with during the optimisation are not actually consistent with a single joint Q. Nevertheless, because we can choose overlapping marginals, the approximation can be a significant improvement on mean field, although it is not in general a lower bound anymore.

Algorithms will maximise the r.h.s. under the constraint that the Q-marginals are at least locally consistent, i.e. marginalise correctly w.r.t. overlaps. On structured graphs, these are known as *loopy belief propagation*, *generalised belief propagation*, etc. [215, 128, 119]. We will be interested in Section 4.3 in a variant of belief propagation called *expectation propagation* [124], but applied to dense rather than sparse graphs.

Maybe the most developed class of approximate inference techniques in Bayesian statistics are *Markov chain Monte Carlo (MCMC)* methods. The idea is that if we were able to obtain independent samples from the target distribution $P(\boldsymbol{\theta})$, we could approximate an expectation w.r.t. P by the corresponding empirical average over the samples. By the law of large numbers, the convergence of this average is roughly $O(n^{-1/2})$, n the number of samples, where the constant depends on smoothness properties of P and the function to average over, but is largely *independent* of the dimensionality of $\boldsymbol{\theta}$. It is usually not possible to directly obtain independent samples from complicated posteriors if $\boldsymbol{\theta}$ is high-dimensional, but it is possible to define an ergodic Markov chain which has the posterior as equilibrium distribution, so that if we run this chain long enough and discard an initial part of the trajectory for "burn-in", the sample path average will converge just as well, again in principle independent of $\boldsymbol{\theta}$'s dimensionality. There is an enormous number of sampling techniques to choose from, for an excellent review consult [130], see also [151, 104, 64]. Powerful software packages for MCMC are available (e.g., [188]).

A.6.4 Lower Bound Maximisation. Expectation Maximisation

A very important concept of classical statistics is *maximum likelihood (ML) estimation*. Suppose we are given some i.i.d. data $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, and we use a model $P(\boldsymbol{x}|\boldsymbol{\theta})$ based on an exponential family (Definition A.5). The log likelihood function is

$$l(\boldsymbol{\theta}) = \log P(S|\boldsymbol{\theta}) = \sum_{i=1}^{n} \log P(\boldsymbol{x}_i|\boldsymbol{\theta}) = n \left(\boldsymbol{\theta}^T \bar{\boldsymbol{\varphi}} - \Phi(\boldsymbol{\theta}) + C \right),$$

where $\bar{\boldsymbol{\varphi}} = n^{-1} \sum_i \boldsymbol{\varphi}(\boldsymbol{x}_i)$ and C is some constant. Note that $l(\boldsymbol{\theta})$ depends on S only via the empirical statistic $\bar{\boldsymbol{\varphi}}$, hence the name "sufficient statistic" for $\boldsymbol{\varphi}$. The unique maximum point of $l(\boldsymbol{\theta})$ is given by the moment parameter $\boldsymbol{\mu} = \bar{\boldsymbol{\varphi}}$: the *maximum likelihood estimate*. It is important to note that $\boldsymbol{\mu} = \bar{\boldsymbol{\varphi}}$ is nothing but the m-projection (moment matching; see Section A.4.2) of the empirical distribution $\hat{P} = n^{-1} \sum_i \delta_{\boldsymbol{x}_i}$ onto the family $\{P(\boldsymbol{x}|\boldsymbol{\theta})\}$. But what if the family is not an exponential one, for example because $P(\boldsymbol{x}|\boldsymbol{\theta}) = \int P(\boldsymbol{x}, \boldsymbol{h}|\boldsymbol{\theta})\, d\boldsymbol{h}$ for some nuisance variables $\boldsymbol{h}$?

This is an instance of the general problem of maximising a function $f(\boldsymbol{\theta}) = \log \int \exp(g(\boldsymbol{h}; \boldsymbol{\theta}))\, d\boldsymbol{h}$. In general, this is a hard task and we may have to content ourselves with finding a local maximum. From (A.11) we know that for fixed $\boldsymbol{\theta}$, f is convex as a function of $g(\cdot; \boldsymbol{\theta})$, so we can find a global lower bound linear in g which is tight at $\boldsymbol{\theta}$:

$$f(\boldsymbol{\theta}') \geq \mathrm{E}_Q \left[g(\boldsymbol{h}; \boldsymbol{\theta}') \right] + \mathrm{H}[Q], \tag{A.19}$$

where $Q(\boldsymbol{h}) \propto \exp(g(\boldsymbol{h}; \boldsymbol{\theta}))$, so that $(g(\cdot; \boldsymbol{\theta}), Q)$ is a Legendre pair. We can now maximise the right hand side w.r.t. $\boldsymbol{\theta}'$. Since for $\boldsymbol{\theta}' = \boldsymbol{\theta}$ both sides are equal, the maximiser will improve on $f(\boldsymbol{\theta})$ if $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \neq \mathbf{0}$. Furthermore, the gradients of $f(\boldsymbol{\theta})$ and the lower bound are identical at $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \mathrm{E}_Q \left[\nabla_{\boldsymbol{\theta}} g(\boldsymbol{h}; \boldsymbol{\theta}) \right]. \tag{A.20}$$

Thus, we are guaranteed convergence to a local maximum of f. The whole procedure relies on $\mathrm{E}_Q \left[g(\boldsymbol{h}; \boldsymbol{\theta}') \right]$ being simple to optimise, while $\int \exp(g(\boldsymbol{h}; \boldsymbol{\theta}))\, d\boldsymbol{h}$ is not. We refer to this method as *lower bound maximisation*. A very important special case is as follows. Let $g(\boldsymbol{h}; \boldsymbol{\theta})$ be the log of an exponential family distribution over $\boldsymbol{h}$ and additional observed variables $\boldsymbol{x}$ (whose values are plugged in) and $\boldsymbol{\theta}$ the natural parameters. The gradient can be computed straightforwardly as $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \mathrm{E}_Q[\boldsymbol{\varphi}(\boldsymbol{h}, \boldsymbol{x})] - \boldsymbol{\mu}$, where $Q(\boldsymbol{h}) = Q(\boldsymbol{h}|\boldsymbol{x})$ is the corresponding conditional distribution (from a different exponential family). The lower bound is maximised by $\boldsymbol{\mu}' = \mathrm{E}_Q[\boldsymbol{\varphi}(\boldsymbol{h}, \boldsymbol{x})]$ which has the same form as the maximum likelihood estimate except that the latent $\boldsymbol{h}$ are marginalised over Q.

The *expectation maximisation (EM)* algorithm for maximum likelihood estimation in the presence of nuisance variables [48] is a special case of this algorithm:

$\boldsymbol{h} = (h_1, \ldots, h_n)$ are nuisance variables, $g(\boldsymbol{h}; \boldsymbol{\theta}) = \sum_i g_i(h_i; \boldsymbol{\theta})$ is the complete data log likelihood which decomposes. Note that the posterior $Q(\boldsymbol{h})$ is a product distribution w.r.t. $h_1, \ldots, h_n$ whose factors are often simple to compute. For many models, the complete data likelihood is actually in an exponential family, so the maximisation of the lower bound simply corresponds to an m-projection (Definition A.8) in the very same way as for ML estimation without nuisance variables (the distribution to be projected is simply the product of the empirical distribution of the observed variables and $Q(\boldsymbol{h})$). Historically, the computation of $Q(\boldsymbol{h})$ is called *E-step*, and the lower bound maximisation is called *M-step*.

Another important interpretation of the EM algorithm comes from information theory. Suppose that the complete data likelihood comes from an exponential family and we observe an instance $\bar{S}$ of the sample variable S. Then, the M-step corresponds to an m-projection, and if we define the manifold $\mathcal{E}$ of distributions over $(S, \boldsymbol{h})$ via the constraint $P(S) = \delta_{\bar{S}}(S),\ P \in \mathcal{E}$, then it is easy to see that the E-step corresponds to an e-projection (Definition A.8) onto $\mathcal{E}$ (the projection results in $\delta_{\bar{S}}(S)Q(\boldsymbol{h})$). EM is a special case of an *alternating minimisation procedure*, a class introduced by Csiszár and Tusnady [43] to show the common roots of a number of algorithms in statistics and information theory. They showed that for some distance $d(P, Q)$, convex in both P and Q, and closed convex sets $\mathcal{P}$, $\mathcal{Q}$, there is a unique global minimum $\min_{P \in \mathcal{P}, Q \in \mathcal{Q}} d(P, Q)$ and a corresponding minimum point (P^*, Q^*) is found by the alternating minimisation procedure. So what's wrong with EM, which can get trapped into local minima? The problem is that the exponential families $\{P(S, \boldsymbol{h}|\boldsymbol{\theta})\}$ commonly used with EM are *not* convex as a set of distributions. Although the m-projections in the M-steps are always well-defined, local minima do exist and are often a real problem. There are simple cases in which EM is used with convex model families, such as the *Blahut-Arimoto* algorithm[10].

There are cases where even the lower bound in (A.19) cannot be computed and optimised feasibly. In this case, we are of course free to choose a different lower bound given by a suboptimal distribution Q. The optimal Q globally maximises

[10] Which computes the maximum likelihood mixing proportions in a mixture model with fixed component distributions, see [34], Sect. 13.8.

the lower bound at $\boldsymbol{\theta}' = \boldsymbol{\theta}$, so our search for Q should do the same thing, but restricted to a family for which the bound can actually be evaluated feasibly. This can be seen as variational generalisation of EM (see Section A.6.3), as was pointed out by Hinton and Neal [75]. Note that this generalised algorithm is not guaranteed to converge to a local minimum of f. It offers a general way of doing marginal likelihood maximisation (see Section A.6.2). There are numerous other generalisations of EM, such as deterministic E-step annealing [153], M-step annealing [167], sequential EM, etc.

A more direct approach to maximising $f(\boldsymbol{\theta})$ is to use gradient-based optimisation. As shown in (A.20), the gradient can be computed at least as easily as maximising the lower bound. Furthermore, gradient-based optimisers such as Quasi-Newton, conjugate gradients or restricted memory Quasi-Newton (e.g., [55]) can exhibit much faster convergence than lower bound maximisation techniques (which typically has first-order convergence). Convergence properties of EM have been studied and compared to those of gradient-based optimisation in [213, 149]. The consensus seems that fairly close to a minimum point, modern gradient-based methods clearly exhibit much faster convergence to high accuracy in $\boldsymbol{\theta}$. On the other hand, EM often attain a point of reasonably low $f(\boldsymbol{\theta})$ significantly faster than gradient-based optimisers, due to its ability to make large steps in $\boldsymbol{\theta}$. Another advantage of EM is that it can often be implemented very easily, although packages for gradient-based optimisation are publicly available. A simple hybrid employed in this thesis uses a modern gradient-based method, but keeps Q fixed during line searches along descent directions. If changes in Q can lead to discontinuous behaviour in the lower bound, this variant promises higher stability.

While EM has its important place amongst modern optimisation techniques if the M-steps can be done efficiently, other lower bound maximisation algorithms are clearly inferior to gradient-based optimisation in many areas of application. An example is the *generalised iterative scaling* algorithm [45] applied to log-linear models for sequence tagging.[11] Here, a sequence of lower bounds is applied to

[11] Iterative scaling is useful for maximum likelihood fitting of undirected graphical models if clique sizes are not very small. However, the improved variant of [190] should be used in which

achieve a decoupling w.r.t. model parameters. While this is quoted as advantage by several authors who have used GIS, it is actually a very questionable thing to do in our opinion. In the optimisation literature, decoupling techniques are avoided because they lead to a phenomenon called *zig-zagging*: given a function which correlates its variables significantly, the optimisation of this function by in turn updating one variable while keeping all others fixed leads to very slow progress in total along a characteristic zig-zag trajectory.

A.7 Large Deviation Inequalities

In this section, we describe Chernoff's bounding technique of deriving exponential inequalities from the basic Markov inequality. We follow the exposition in [114].

Theorem A.1 (Markov Inequality) *Let X be an almost surely non-negative random variable with finite mean. Then, for every $\varepsilon > 0$,*

$$\Pr\{X \geq \varepsilon\} \leq \frac{\mathrm{E}[X]}{\varepsilon}.$$

For the proof, assume w.l.o.g. that $\varepsilon = 1$ and note that $\Pr\{X \geq 1\} = \mathrm{E}[\mathrm{I}_{\{X \geq 1\}}] \leq \mathrm{E}[X]$. *Chebishev's inequality* for a variable Y with zero mean and finite variance,

$$\Pr\{|Y| \geq \varepsilon\} \leq \frac{\mathrm{Var}[Y]}{\varepsilon^2}$$

follows from Markov's by setting $X = Y^2$.

However, in order to bound probabilities of events which are exponentially unlikely, we require inequalities which are exponential in nature. A standard technique to obtain tighter large deviation results is to apply Markov's inequality to the *exponentiated* large deviation. The technique is often accredited to Chernoff [28], but has actually been used by Cramér before [35]:

$$\Pr\{X \geq x\} = \Pr\left\{e^{\beta X} \geq e^{\beta x}\right\} \leq e^{-\beta x}\mathrm{E}\left[e^{\beta X}\right], \quad x \geq \mathrm{E}[X].$$

The same inequality holds for $\Pr\{X \leq x\}$, $x \leq \mathrm{E}[X]$. In the remainder of this section, we will focus on the upper tail.

parameter updates and inference are interleaved.

Theorem A.2 (Chernoff Inequality) *Let X be a random variable with finite mean. Then, for every $x \geq \mathrm{E}[X]$,*

$$\Pr\{X \geq x\} \leq e^{-S(X,x)}, \quad S(X,x) = \sup_{\beta} x\beta - \log \mathrm{E}\left[e^{\beta X}\right].$$

Let $K(\beta) = \log \mathrm{E}[e^{\beta X}]$ be the *cumulant generating function* of X. By definition, the cumulants of X (if they exist) can be obtained as derivatives of $K(\beta)$, evaluated at $\beta = 0$. Note that this implies that $K(\beta)$ is convex w.r.t. β, and therefore its convex dual $S(X,x)$ is convex w.r.t. x (see Section A.3). If P is the distribution of X, we can define the tilted distribution P_β by $dP_\beta(X) = \exp(\beta X - K(\beta))\, dP(X)$ whenever $K(\beta) < \infty$. It is easy to see that $S(X,x) = \mathrm{D}[P_{\beta(x)} \,\|\, P]$, where $(x, \beta(x))$ is a Legendre pair (see Section A.3). Since the cumulant generating function of P_β is $K_\beta(\gamma) = K(\gamma + \beta) - K(\beta)$, the cumulants of P_β are given by the derivatives of $K(\beta)$ w.r.t. β, evaluated at β.

If $K(\beta)$ is known, then $S(X,x)$ can be computed by noting that the supremum is attained at $\beta = \beta(x)$ such that $K'(\beta) = \mathrm{E}_\beta[X] = x$. If $X = \sum_{i=1}^n X_i$, with the X_i being distributed i.i.d., we have that $K(\beta) = n\, K_1(\beta) = n \log \mathrm{E}[e^{\beta X_1}]$. In this case, we look for β such that $K_1'(\beta) = x/n$, then $S(X,x) = n(\beta(x/n) - K_1(\beta))$. Cramér has shown that the bound of Theorem A.2 is asymptotically tight in this case, i.e. for every fixed $q \in [0,1]$:

$$\lim_{n\to\infty} \frac{1}{n} \left(\log \Pr\{X \geq n\, q\} + S(X, n\, q)\right) = 0.$$

For example, if the X_i are i.i.d. Bernoulli(p) variables and $x \in [0,n]$, then $K_1(\beta) = \log(1 - p + p\, e^\beta)$ and $S(X,x) = n\, \mathrm{D}_{\mathrm{Ber}}[x/n \,\|\, p]$ (see Equation A.7).

Theorem A.3 (Chernoff Inequality, Binomial Variables) *Let $X_1, \ldots, X_n$ be i.i.d. Bernoulli(p) variables and $X = \sum_{i=1}^n X_i$. Then, for every $q \in [p,1]$,*

$$\Pr\{X \geq n\, q\} \leq e^{-n\, \mathrm{D}_{\mathrm{Ber}}[q \,\|\, p]}.$$

The same bound holds for $\{X \leq n\, q\}$, $q \leq p$. Furthermore, for $\varepsilon > 0$,

$$\Pr\{\mathrm{D}_{\mathrm{Ber}}[X/n \,\|\, p] \geq \varepsilon,\ X/n < p\} \leq e^{-n\,\varepsilon},$$

and the same holds for $\{\mathrm{D}_{\mathrm{Ber}}[X/n \,\|\, p] \geq \varepsilon,\ X/n > p\}$. *These bounds hold also if* $X_1, \ldots, X_n$ *are i.i.d. bounded variables,* $X_i \in [0,1]$, $\mathrm{E}[X_i] = p$. *However, they are tightest for sums of Bernoulli variables.*

Namely, let $X_i \in [0,1]$ and $\mathrm{E}[X_i] = p$. Since $e^{\beta X_1}$ is convex, we have $e^{\beta X_1} \leq 1 - X_1 + X_1\, e^{\beta}$ for $X_1 \in [0,1]$, thus $K_1(\beta) \leq \tilde{K}_1(\beta)$, where $\tilde{K}_1$ is the cumulant generating function for a Bernoulli(p) variable. Therefore, $S(X,x) \geq S(\tilde{X},x)$, where $\tilde{X}$ is the sum of n i.i.d. Bernoulli(p) variables. If the appearance of $\mathrm{D}_{\mathrm{Ber}}$ is inconvenient, we can use the quadratic lower bound $\mathrm{D}_{\mathrm{Ber}}[q \,\|\, p] \geq 2(q-p)^2$ in order to derive *Hoeffding's Inequality*:

$$\Pr\{X/n \geq p + \varepsilon\} \leq e^{-2n\,\varepsilon^2}.$$

However, the quadratic bound is fairly poor if X/n is far from $1/2$ (which is usually what we are interested in), and the use of Hoeffding's inequality "just for convenience" is not recommended.

Appendix B

Appendix for Chapter 3

In this chapter, we provide derivations and additional details in order to complete the presentation in Chapter 4.

B.1 Extended PAC-Bayesian Theorem: an Example

Theorem 3.2 provides a powerful extension of the PAC-Bayesian Theorem 3.1 to multiple classes and confusion distribution. In this section, we work through an example in order to see how it can be used in practice. In the binary classification case, we might be interested in upper bounding the probability that the Gibbs rule outputs $\tilde{y} = +1$, given that the true class is $y_* = -1$ (i.e. the probability of a false positive). This probability can be written as $F(y_*, \tilde{y})/P(y_*)$, where $F(y_*, \tilde{y})$ is the joint confusion distribution of (3.4).

We first use Chernoff's inequality (Theorem A.2) to obtain a lower bound on $p_* = P(y_*)$. For $\hat{p}_* = n^{-1} \sum_i \mathrm{I}_{\{y_i = y_*\}}$,

$$\Pr_S \left\{ \mathrm{D}_{\mathrm{Ber}}[\hat{p}_* \,\|\, p_*] > \varepsilon,\ \hat{p}_* > p_* \right\} \leq e^{-n\,\varepsilon}.$$

Thus, if ν is chosen such that $\mathrm{D}_{\mathrm{Ber}}[\hat{p}_* \,\|\, \hat{p}_* - \nu] = \varepsilon = (1/n) \log \delta_1^{-1}$, then we have $p_* \geq \hat{p}_* - \nu$ with probability at least $1 - \delta_1$. Note that $\hat{p}_* - \nu$ is the left limit of the interval $\mathrm{D}_{\mathrm{Ber}}^{-1}(\hat{p}_*, (1/n) \log \delta_1^{-1})$, as defined in (3.1). We will show in a moment how to use Theorem 3.2 in order to obtain an upper bound on $F(y_*, \tilde{y})$ which holds with confidence $1 - \delta_2$. Applying the union bound, we conclude that *both*

bounds hold at the same time with confidence $1 - \delta_1 - \delta_2$, resulting in an upper bound on the ratio $F(y_*, \tilde{y})/P(y_*)$. The confidence parameters must be chosen *a priori*. Since the bound on $P(y_*)$ is much tighter, you may want to choose $\delta_1 < \delta_2$.

We now apply Theorem 3.2 using the setup for $\mathcal{L}$ and l introduced after (3.4). Note that $L = 4$ and $\boldsymbol{p} = \boldsymbol{p}(Q) = (p_1 \dots p_4)$. W.l.o.g. let p_1 be the entry representing $F(y_*, \tilde{y})$ for the values $y_* = +1$, $\tilde{y} = -1$. Writing $\hat{\boldsymbol{p}} = \hat{\boldsymbol{p}}(S, Q)$ and $\varepsilon = \varepsilon(\delta, n, P, Q)$, we can upper bound $F(y_*, \tilde{y})$ by the maximum value of p_1, given that $\boldsymbol{p} \in \mathcal{P}(\hat{\boldsymbol{p}}, \varepsilon)$, i.e. lies in the convex bounded feasible set defined in (3.6). This is an optimisation problem with linear criterion and convex constraints which can be solved easily, at least under the assumption that all components of $\hat{\boldsymbol{p}}$ are positive. It is shown below in this subsection that the solution has the form

$$p_1 = \sqrt{\lambda \hat{p}_1 + \frac{1}{4}(\lambda - 1)^2} - \frac{1}{2}(\lambda - 1), \quad p_l = \frac{\lambda}{\lambda + p_1} \hat{p}_l, \; l = 2, 3, 4. \tag{B.1}$$

Here, $\lambda > 0$ is a Lagrange multiplier. If we denote the solution for a fixed multiplier by $\boldsymbol{p}_\lambda$, then the function $\lambda \mapsto \mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}_\lambda]$ is strictly monotonically decreasing, thus a simple one-dimensional search will find the unique multiplier value λ for which $\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}_\lambda] = \varepsilon$. The solution is $\boldsymbol{p} = \boldsymbol{p}_\lambda$ for this value of λ, and its component p_1 is the desired upper bound for $F(y_*, \tilde{y})$.

It remains to show that (B.1) is the solution for the convex problem stated above. This is a straightforward exercise in using Lagrange multipliers. The primal problem is to minimise $-p_1 = -\boldsymbol{\delta}_1^T \boldsymbol{p}$, subject to $\boldsymbol{p} \in \mathcal{P}(\hat{\boldsymbol{p}}, \varepsilon)$, where the feasible set $\mathcal{P}(\hat{\boldsymbol{p}}, \varepsilon)$ is given by (3.6). We assume that all components of $\hat{\boldsymbol{p}}$ are positive. Let us introduce Lagrange multipliers $\lambda \geq 0$ and ρ for the constraints $\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}] - \varepsilon \leq 0$ and $\mathbf{1}^T \boldsymbol{p} - 1 = 0$. The Lagrangian is

$$\Lambda = -\boldsymbol{\delta}_1^T \boldsymbol{p} + \lambda(\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}] - \varepsilon) + \rho(\mathbf{1}^T \boldsymbol{p} - 1).$$

Setting the gradient of Λ w.r.t. $\boldsymbol{p}$ equal to zero, we obtain

$$-\delta_{l,1} - \lambda \frac{\hat{p}_l}{p_l} + \rho = 0, \quad l = 1, \dots, L.$$

Since all $\hat{p}_l > 0$, all p_l must be positive as well, otherwise $\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}] = \infty$ would

violate the constraint. Thus,

$$p_l(\rho - \delta_{l,1}) = \lambda \hat{p}_l, \quad l = 1, \ldots, L.$$

Summing over l and using the constraints, we see that $\rho = \lambda + p_1$ and

$$p_l = \frac{\lambda}{\lambda + p_1 - \delta_{l,1}} \hat{p}_l.$$

For $l = 1$, this is a quadratic in p_1 with the unique nonnegative solution

$$p_1 = \sqrt{\lambda \hat{p}_1 + \frac{1}{4}(\lambda - 1)^2} - \frac{1}{2}(\lambda - 1).$$

It is easy to see that $dp_1/d\lambda < 0$ for all $\lambda > 0$, with $p_1 = 1$ for $\lambda = 0$ and $p_1 \to \hat{p}_1$ as $\lambda \to \infty$. If $\boldsymbol{p}_\lambda = (p_{\lambda,l})_l$,

$$p_{\lambda,1} = \sqrt{\lambda \hat{p}_1 + \frac{1}{4}(\lambda - 1)^2} - \frac{1}{2}(\lambda - 1), \quad p_{\lambda,l} = \frac{\lambda}{\lambda + p_{\lambda,1}} \hat{p}_l, \; l > 1,$$

then $\lambda \mapsto \mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}_\lambda]$ is strictly decreasing, and $\boldsymbol{p}_\lambda \to \boldsymbol{\delta}_1$ $(\lambda \to 0)$, $\boldsymbol{p}_\lambda \to \hat{\boldsymbol{p}}$ $(\lambda \to \infty)$. Thus, a simple one-dimensional search reveals the unique $\lambda > 0$ such that $\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}_\lambda] = \varepsilon$, and $\boldsymbol{p} = \boldsymbol{p}_\lambda$ is the solution we are looking for.

B.2 Details of Proof of Theorem 3.2

Here, we give the proof of Inequality (3.7) (see [34], Sect. 12.1). Recall the notation defined in the proof of Theorem 3.2 (Section 3.2.2). First, the probability of a sequence $(l_1, \ldots, l_n)$ depends on its type only (i.e. the type is a sufficient statistic for the sequence):

$$\prod_{i=1}^{n} p_{l_i} = e^{n(-\mathrm{D}[\hat{\boldsymbol{p}} \,\|\, \boldsymbol{p}] - \mathrm{H}[\hat{\boldsymbol{p}}])} \tag{B.2}$$

(recall the entropy $\mathrm{H}[\hat{\boldsymbol{p}}]$ from Equation A.9). Second, the number of sequences with type $\hat{\boldsymbol{p}}$ can be upper bounded by $e^{n\mathrm{H}[\hat{\boldsymbol{p}}]}$. To see this, consider sequences being sampled i.i.d. from $\hat{\boldsymbol{p}}$ (not from $\boldsymbol{p}$). Under this distribution, the probability of an arbitrary sequence of type $\hat{\boldsymbol{p}}$ is $e^{-n\mathrm{H}[\hat{\boldsymbol{p}}]}$ (using (B.2)). Since the sum of these probabilities must be ≤ 1, we obtain the bound on the number of sequences. Note

that this bound is very tight: a lower bound on this number is $e^{n\mathrm{H}[\hat{\boldsymbol{p}}]}\,(n+1)^{1-L}$ (see [34], Theorem 12.1.3). Third, the number of different types is upper bounded by $(n+1)^{L-1}$. Thus, we have

$$\mathrm{E}_S\left[e^{n\mathrm{D}[\hat{\boldsymbol{p}}\,\|\,\boldsymbol{p}]}\right] \leq \sum_{\hat{\boldsymbol{p}}} e^{n\mathrm{D}[\hat{\boldsymbol{p}}\,\|\,\boldsymbol{p}]} e^{n(-\mathrm{D}[\hat{\boldsymbol{p}}\,\|\,\boldsymbol{p}]-\mathrm{H}[\hat{\boldsymbol{p}}])} e^{n\mathrm{H}[\hat{\boldsymbol{p}}]} = \sum_{\hat{\boldsymbol{p}}} 1 \leq (n+1)^{L-1}.$$

By using the lower bound on the number of sequences of given type, it is easy to show that $\mathrm{E}_S[\exp(n\,\lambda\,\mathrm{D}[\hat{\boldsymbol{p}}\,\|\,\boldsymbol{p}])]$ grows exponentially in n for every $\lambda > 1$, so that in this sense the exponent we are using is optimal.

B.3 Proof of Theorem 3.3

Here, we give a proof of the PAC-Bayesian theorem for arbitrary bounded loss functions, Theorem 3.3. This is done by starting from an inequality bounding the probability of large deviations between expected and empirical loss for fixed rules in order to obtain a "exponential" statement like (3.8), which can then be plugged into the generic part of the proof as given in Section 3.2.2. This implication is proved using a beautiful "water-filling" argument due to [117].

Recall the notation defined in Sections 3.2.4 and 3.2.2. Given a loss function $l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_*, y_*)) \in [0,1]$, the risk is defined as $l(\tilde{\boldsymbol{w}}) = \mathrm{E}[l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_*, y_*))]$ and the empirical risk as $\hat{l}(\tilde{\boldsymbol{w}}) = n^{-1}\sum_i l(\tilde{\boldsymbol{w}}, (\boldsymbol{x}_i, y_i))$. For some nonnegative function ϕ on $[0,1]^2$, we assume that the inequalities (3.17) hold for every $\tilde{\boldsymbol{w}}$, moreover that $\phi(q,l)$ is nondecreasing in $|q-l|$, convex in (q,l), $\phi(l,l) = 0$ for all $l \in [0,1]$, finally that $\phi(\cdot, l)$ is differentiable on $(0,l)$ and $(l,1)$, for all $l \in (0,1)$. Now fix $\tilde{\boldsymbol{w}}$ and let $l = l(\tilde{\boldsymbol{w}})$, $\hat{l} = \hat{l}(\tilde{\boldsymbol{w}})$. If we can show that

$$\mathrm{E}_S\left[e^{(n-1)\phi(\hat{l},l)}\right] \leq 2n+1, \tag{B.3}$$

we can take the expectation over $\tilde{\boldsymbol{w}} \sim P$, exchange the order of the two expectations and apply Markov's inequality to end up with the statement we are after:

$$\Pr_S\left\{\mathrm{E}_{\tilde{\boldsymbol{w}}\sim P}\left[e^{(n-1)\phi(\hat{l}(\tilde{\boldsymbol{w}}),l(\tilde{\boldsymbol{w}}))}\right] > \frac{2n+1}{\delta}\right\} \leq \delta.$$

Note that if $l \in \{0, 1\}$, then $\hat{l} = l$ with probability 1, thus (B.3) is trivially true. In order to prove (B.3) for $l \in (0, 1)$, we look for the distribution over $\hat{l}$ which maximises the left-hand side of (B.3), subject to the constraints imposed by (3.17) for every q. Since $\phi(q, l)$ is nondecreasing in $|q - l|$, it is obvious that a maximiser should fulfil these constraints with equality for all[1] q. A measure over $\hat{l}$ with density $f(\hat{l})$ fulfilling all constraints with equality can be obtained by differentiating the constraints w.r.t. q. We obtain

$$f(\hat{l}) = \operatorname{sgn}(l - \hat{l}) \frac{d}{d\hat{l}} e^{-n\phi(\hat{l},l)}.$$

It is straightforward to check that $f(\hat{l})$ is nonnegative and integrates to $2 - e^{-n\phi(0,l)} - e^{-n\phi(1,l)} \leq 2$. Since this may be a value < 1, we consider the measure over $\hat{l}$ which is the sum of the measure with density $f(\hat{l})$ and a point measure of mass 1 at l. The expectation of $e^{(n-1)\phi(\hat{l},l)}$ over this measure is an upper bound on this expectation over any probability distribution fulfilling the constraints (3.17), thus:

$$\begin{aligned}
\mathrm{E}\left[e^{(n-1)\phi(\hat{l},l)}\right] &\leq 1 + \int_0^1 e^{(n-1)\phi(\hat{l},l)} \operatorname{sgn}(l - \hat{l}) \frac{d}{d\hat{l}} e^{-n\phi(\hat{l},l)} \, d\hat{l} \\
&= 1 + n \int_0^1 \operatorname{sgn}(\hat{l} - l) \frac{d\phi}{d\hat{l}}(\hat{l}, l) e^{-\phi(\hat{l},l)} \, d\hat{l} \\
&= 1 + n \int_0^1 \operatorname{sgn}(l - \hat{l}) \frac{d}{d\hat{l}} e^{-\phi(\hat{l},l)} \, d\hat{l} \\
&= 1 + n \left(2 - e^{-\phi(0,l)} - e^{-\phi(1,l)}\right) \leq 2n + 1,
\end{aligned}$$

proving the statement (B.3). Applying the generic part of the proof in Section 3.2.2 and using the convexity of $\phi(q, l)$ completes the proof of Theorem 3.3.

[1] This is a *water-filling* argument. The density function $f(\hat{l})$ fulfilling all constraints with equality may integrate to more than 1. In this case, the maximising probability density is obtained by shrinking the initial part of $f(\hat{l})$ to zero. Since this tightens the inequality only insignificantly, we however stick with $f(\hat{l})$ here.

B.4 Efficient Evaluation of the Laplace GP Gibbs Classifier

Recall from Section 2.1.3 that in order to evaluate the Laplace GP Gibbs classifier on a test point $\boldsymbol{x}_*$, we require the predictive variance $\sigma^2(\boldsymbol{x}_*)$ which can be computed in $O(n^2)$ using a back-substitution. The same is true for all other non-sparse members of the family of approximation methods mentioned in Section 2.1.3 (i.e., $d = n$). Note that $\sigma^2(\boldsymbol{x}_*)$ is required in order to compute an uncertainty estimate for both Bayes and Gibbs classifier, but if this is not required, the rejection sampling technique developed here can be used to end up with more efficient predictions. We describe this technique for the case of Laplace GPC, but the development for any other non-sparse method should be clear using the notation introduced in Sections 3.3.1 and 2.1.3.

Fix $\boldsymbol{x}_*$ and let $\boldsymbol{b} = \boldsymbol{D}^{1/2}\boldsymbol{k}(\boldsymbol{x}_*)$. Then,

$$\sigma_*^2 = \sigma^2(\boldsymbol{x}_*) = K(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{b}^T \boldsymbol{B}^{-1} \boldsymbol{b}.$$

It is possible to obtain upper and lower bounds on the critical term $\boldsymbol{b}^T \boldsymbol{B}^{-1} \boldsymbol{b}$ *without* having to compute a factorisation of $\boldsymbol{B}$, by using Lanczos methods (e.g., [157]). The bounds are due to Skilling [178] and have been used in the context of GP methods by [63]. The idea is to maximise the quadratic function

$$q(\boldsymbol{u}) = \boldsymbol{b}^T \boldsymbol{u} - \frac{1}{2} \boldsymbol{u}^T \boldsymbol{B} \boldsymbol{u}$$

approximately. If we do a *sparse greedy* maximisation of q, allowing only a controlled number d of components of $\boldsymbol{u}$ to become nonzero, we can compute upper and lower bounds on σ_*^2 in $O(n\, d^2)$ (see [183]), without having to know more than d rows of $\boldsymbol{B}$. Alternatively, faster incomplete Cholesky techniques can be used [54]. Now suppose we are given $\sigma_L^2 \leq \sigma_*^2 \leq \sigma_U^2$. Then, we can create envelope functions $L(u_*)$ and $U(u_*)$ on the p.d.f. $f(u_*) = N(u_*|0, \sigma_*^2)$, simply by setting $L(u_*) = (2\pi\sigma_*^2)^{-1/2} \exp(-u_*^2/(2\sigma_L^2))$ and $U(u_*) = (2\pi\sigma_*^2)^{-1/2} \exp(-u_*^2/(2\sigma_U^2))$. Note that $U(u_*) \propto g(u_*) = N(u_*|0, \sigma_U^2)$, and that we can compute the ratio

$$r(u_*) = \frac{L(u_*)}{U(u_*)} = \exp\left(-\frac{u_*^2}{2}\left(\frac{1}{\sigma_L^2} - \frac{1}{\sigma_U^2}\right)\right) \tag{B.4}$$

without knowledge of σ_*^2. We can use these facts to sample $u_* \sim N(0, \sigma_*^2)$, using a sandwiched rejection method (e.g., [150]), as follows:

- Independently sample $u_* \sim N(0, \sigma_U^2)$ and u uniformly from $[0, 1]$.
- If $u \leq r(u_*)$, return u_*. The ratio $r(u_*)$ is given by (B.4).
- Otherwise, compute the variance σ_*^2 exactly. Now, if $u \leq f(u_*)/U(u_*)$, return u_*. Otherwise, sample $u_* \sim N(0, \sigma_*^2)$ and return it.

Note that the method produces a sample of $f(u_*)$ *without* having to compute the variance σ_*^2, whenever the algorithm returns in the second step. Let E denote this event, and denote $V(U) = \int U(u_*)\, du_*$, $V(L) = \int L(u_*)\, du_*$. Now,

$$\begin{aligned} Pr\{E\} &= Pr\left\{u \leq \frac{L(u_*)}{U(u_*)}\right\} = E\left[\frac{L(u_*)}{U(u_*)}\right] = \int \frac{L(u_*)}{V(U)g(u_*)} g(u_*)\, du_* \\ &= \frac{V(L)}{V(U)} = \frac{\sigma_L}{\sigma_U}, \end{aligned}$$

where we have used that $U(u_*) = V(U)g(u_*)$. If the bounds are tight, this probability is close to 1. For example, if we have $\sigma_L^2 \geq (1-\varepsilon)\sigma_*^2$, $\sigma_U^2 \leq (1+\varepsilon)\sigma_*^2$, then

$$V(L)/V(U) \geq \sqrt{(1-\varepsilon)/(1+\varepsilon)} = 1 - \varepsilon/(1+\varepsilon) + O(\varepsilon^2).$$

B.5 Proof of Theorem 3.4

Theorem 3.4 is a PAC-Bayesian theorem for Bayes voting classifiers. In this section, we replicate the proof given in [121] in our notation. Section 2.2 contains an introduction to some of the concepts used here.

The proof uses a recent bound given in [91]. We need some machinery from empirical process theory. Let $\mathcal{F}$ be a set of classifiers[2] $f : \mathcal{X} \to \{-1, +1\}$. Let $S = \{(\boldsymbol{x}_i, y_i)\}$ be an n-sample as above, and define the *empirical Rademacher complexity* of $\mathcal{F}$ as

$$\hat{R}_n(\mathcal{F}) = \mathrm{E}_{\boldsymbol{\varepsilon}}\left[\sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i f(\boldsymbol{x}_i) \right|\right], \tag{B.5}$$

[2]Of course, there are some measurability issues here, we refer to the original work.

where $\boldsymbol{\varepsilon}$ is a vector of i.i.d. Rademacher variables (unbiased coins with sides $+1, -1$), independent of S. Note that $\hat{R}_n(\mathcal{F})$ is a random variable, but independent of the targets y_i. It can be shown that it is concentrated around $R_n(\mathcal{F}) = \mathrm{E}_S[\hat{R}_n(\mathcal{F})]$, the *Rademacher complexity* of $\mathcal{F}$, if S is drawn i.i.d. as usual (in fact, $\hat{R}_n(\mathcal{F})$ has bounded differences of size $\leq 1/n$, so Hoeffding's inequality applies, see Section 2.2.2). Let ϕ be the margin loss function defined in Section 3.4.1, and choose $\delta \in (0,1)$, $c > 1$.

Theorem B.1 ([91], Theorem 2) *For any data distribution over $\mathcal{X} \times \{-1,+1\}$, we have that*

$$\Pr_S \left\{ \Pr_{(\boldsymbol{x}_*, y_*)} \{ y_* f(\boldsymbol{x}_*) \leq 0 \} > \inf_{\kappa \in (0,1]} B(\kappa, S, Q) + \sqrt{\frac{\log(2/\delta)}{2n}} \text{ for some } f \in \mathcal{F} \right\} \leq \delta.$$

Here,

$$B(\kappa, S, Q) = \frac{1}{n} \sum_{i=1}^{n} \phi\left(y_i f(\boldsymbol{x}_i)/\kappa\right) + \frac{4\,c}{\kappa} R_n(\mathcal{F}) + \frac{\log\log(2/\kappa) - \log\log c}{n}.$$

The proof of this theorem is not very hard. First, the bounded loss allows us to apply Hoeffding's inequality based on bounded Martingale difference sequences (see Section 2.2.2). We then use Ledoux and Talagrands contraction principle to get rid of the margin loss function, exploiting the fact that $(\phi(\cdot/\kappa) - 1)/\kappa$ is a contraction.[3] The details are in [91]. Recall the definitions of $f(\boldsymbol{x}|Q)$, $\mathcal{Q}_A$ and $\mathcal{F}_A$ from Section 3.4.1. We can use convex duality for $\mathrm{D}[Q \,\|\, P]$ in order to bound the empirical Rademacher complexity $\hat{R}_n(\mathcal{F}_A)$, getting rid of the supremum over $\mathcal{F}_A$. First note that since every class $\mathcal{F}_A$ is closed under multiplication with -1, we can remove the absolute value operator in (B.5). Recall that for any (measurable) $z(\boldsymbol{w})$,

$$\mathrm{E}_{\boldsymbol{w} \sim Q}[z(\boldsymbol{w})] \leq \mathrm{D}[Q \,\|\, P] + \log \mathrm{E}_{\boldsymbol{w} \sim P}\left[e^{z(\boldsymbol{w})}\right]$$

[3] f is a contraction if $|f(x) - f(y)| \leq |x - y|$ for all x, y.

(see Equation A.8). Write short $y^{(i)} = y(\boldsymbol{x}_i|\boldsymbol{w})$. Choosing $z(\boldsymbol{w}) = (\lambda/n)\sum_i \varepsilon_i y^{(i)}$, we have

$$\hat{R}_n(\mathcal{F}_A) \leq \frac{1}{\lambda}\left(A + \mathrm{E}_{\boldsymbol{\varepsilon}}\left[\log \mathrm{E}_{\boldsymbol{w}\sim P}\left[\exp\left(\frac{\lambda}{n}\sum_i \varepsilon_i y^{(i)}\right)\right]\right]\right). \tag{B.6}$$

Now use Jensen's inequality to pull the expectation over $\boldsymbol{\varepsilon}$ inside the log and evaluate the moment generating function of the ε_i:

$$\mathrm{E}_{\boldsymbol{\varepsilon}}\left[\log \mathrm{E}_{\boldsymbol{w}\sim P}\left[\exp\left(\frac{\lambda}{n}\sum_i \varepsilon_i y^{(i)}\right)\right]\right] \leq \log \mathrm{E}_{\boldsymbol{w}\sim P}\left[\prod_{i=1}^n \cosh\left(\frac{\lambda\, y^{(i)}}{n}\right)\right].$$

Note that $\log\cosh x$ is concave as a function of x^2 (e.g., [80], Sect. 3.B), thus can be upper bounded by quadratics. We use $\log\cosh x \leq (1/2)x^2$:

$$\log \mathrm{E}_{\boldsymbol{w}\sim P}\left[\prod_{i=1}^n \cosh(\lambda\, y^{(i)}/n)\right] \leq \log \mathrm{E}_{\boldsymbol{w}\sim P}\left[\exp\left(\frac{\lambda^2}{2\,n^2}\sum_i (y^{(i)})^2\right)\right] = \frac{\lambda^2}{2\,n},$$

and minimisation w.r.t. λ leads to

$$\hat{R}_n(\mathcal{F}_A) \leq \sqrt{\frac{2\,A}{n}},$$

the same bound holds for $R_n(\mathcal{F}_A)$. For fixed A, Theorem B.1 holds with $\mathcal{F} = \mathcal{F}_A$ and $R_n(\mathcal{F})$ replaced by $\sqrt{2\,A/n}$. Now, replace A and δ by A_j and $\delta_j = p_j\delta$. Since $\sum_j \delta_j = \delta$, we can conclude the proof of Theorem 3.4 by applying the union bound.

B.5.1 The Case of Regression

In Section 3.4.1.2, we mentioned a possible route towards a PAC-Bayesian theorem for GP regression. Here, we give the details for this argumentation. Our starting point is Theorem 1 in [91], where in the second part they prove a bound on $\mathrm{E}\|P_n - P\|_{\mathcal{G}_\phi}$ in terms of the Rademacher complexity $R_n(\mathcal{F}_A)$. In the case we are interested in, $\mathcal{G}_\phi$ consists of the functions $(\boldsymbol{x}, y) \mapsto \phi(y - f(\boldsymbol{x}))$, $f \in \mathcal{F}_A$. Recall that $f(\boldsymbol{x}) = \mathrm{E}_Q[u(\boldsymbol{x}|\boldsymbol{w})]$, $Q \in \mathcal{Q}_A$, and that ϕ is a Lipschitz-κ loss function with $\phi(0) = 0$. Obviously, this part of the proof remains valid, and we have

$$\mathrm{E}\|P_n - P\|_{\mathcal{G}_\phi} \leq \frac{4}{\kappa} R_n(\mathcal{F}_A),$$

where the Rademacher complexity is now given by $R_n(\mathcal{F}_A) = E_S[\hat{R}_n(\mathcal{F}_A)]$,

$$\hat{R}_n(\mathcal{F}_A) = \mathrm{E}_{\boldsymbol{\varepsilon}}\left[\sup_{f\in\mathcal{F}_A, s\in\{-1,+1\}} \frac{s}{n}\sum_{i=1}^{n}\varepsilon_i(y_i - f(\boldsymbol{x}_i))\right].$$

Denote $u_i = u(\boldsymbol{x}_i|\boldsymbol{w})$, so that $f(\boldsymbol{x}_i) = \mathrm{E}_Q[u_i]$. We can use convex duality as above, with $z(\boldsymbol{w}) = (\lambda/n)\sum_i \varepsilon_i(y_i - u_i)$, resulting in

$$\hat{R}_n(\mathcal{F}_A) \leq \frac{1}{\lambda}\left(A + \mathrm{E}_{\boldsymbol{\varepsilon}}\left[\max_{s\in\{-1,+1\}} \log \mathrm{E}_P\left[\exp\left(\frac{\lambda\, s}{n}\boldsymbol{\varepsilon}^T(\boldsymbol{y}-\boldsymbol{u})\right)\right]\right]\right).$$

Note that $\boldsymbol{u} \sim N(\boldsymbol{0}, \boldsymbol{K})$. Setting $r = -s\,\boldsymbol{\varepsilon}^T\boldsymbol{u} \sim N(0, \boldsymbol{\varepsilon}^T\boldsymbol{K}\boldsymbol{\varepsilon})$, we have

$$\begin{aligned}\log \mathrm{E}_P\left[\exp\left(\frac{\lambda\, s}{n}\boldsymbol{\varepsilon}^T(\boldsymbol{y}-\boldsymbol{u})\right)\right] &= \frac{\lambda\, s}{n}\boldsymbol{\varepsilon}^T\boldsymbol{y} + \log \mathrm{E}_P\left[e^{\lambda\, r/n}\right] \\ &= \frac{\lambda\, s}{n}\boldsymbol{\varepsilon}^T\boldsymbol{y} + \frac{\lambda^2}{2\,n^2}\boldsymbol{\varepsilon}^T\boldsymbol{K}\boldsymbol{\varepsilon}.\end{aligned}$$

All in all, if

$$\Psi(\boldsymbol{y}) = \mathrm{E}_{\boldsymbol{\varepsilon}}\left[\max_{s\in\{-1,+1\}} s\,\boldsymbol{\varepsilon}^T\boldsymbol{y}\right] = \mathrm{E}_{\boldsymbol{\varepsilon}}\left[\left|\boldsymbol{\varepsilon}^T\boldsymbol{y}\right|\right],$$

we have

$$\mathrm{E}_{\boldsymbol{\varepsilon}}\left[\max_{s\in\{-1,+1\}} \log \mathrm{E}_P\left[\exp\left(\frac{\lambda\, s}{n}\boldsymbol{\varepsilon}^T(\boldsymbol{y}-\boldsymbol{u})\right)\right]\right] = \frac{\lambda^2}{2\,n^2}\operatorname{tr}\boldsymbol{K} + \frac{\lambda}{n}\Psi(\boldsymbol{y}),$$

where we used $\mathrm{E}[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T] = \boldsymbol{I}$. A minimisation w.r.t. λ results in

$$\hat{R}_n(\mathcal{F}_A) \leq n^{-1/2}\sqrt{2\,A\,(\operatorname{tr}\boldsymbol{K}/n)} + n^{-1}\Psi(\boldsymbol{y}).$$

Finally, making use of Jensen's inequality on the concave square root function, we have

$$\mathrm{E}_S[\Psi(\boldsymbol{y})] = \mathrm{E}_{\boldsymbol{\varepsilon},S}\left[\sqrt{(\boldsymbol{\varepsilon}^T\boldsymbol{y})^2}\right] \leq \sqrt{\mathrm{E}\left[(\boldsymbol{\varepsilon}^T\boldsymbol{y})^2\right]} = \sqrt{\mathrm{E}\left[\boldsymbol{y}^T\boldsymbol{y}\right]} = \sqrt{n\,\mathrm{E}\left[y^2\right]},$$

and (3.24) follows easily.

B.6 Proof of a PAC Compression Bound

In Section 3.5.4, we empirically compare the PAC-Bayesian Theorem 3.1 for binary GP classification with a standard PAC compression bound. In this section,

we state this bound (Theorem B.2) and give a proof. The theorem here is a slight refinement of Theorem 5.18 in [72] which in turn is based on results in [103]. It is interesting to note that in spite of the extremely simple derivation of the compression bound, it seems to be among the tightest known bounds for the popular support vector machine in practically relevant regimes, way ahead of many results employing much deeper mathematics.

Let us define what we mean by a compression scheme. Suppose we are given a hypothesis space $\mathcal{H}$ of binary classifiers and a learning algorithm A mapping i.i.d. training samples S from an unknown data distribution to hypotheses $A(S) \in \mathcal{H}$. As usual, we define generalisation and empirical error of a hypothesis h as

$$\mathrm{gen}(h) = \Pr_{(\boldsymbol{x}_*, y_*)}\{h(\boldsymbol{x}_*) \neq y_*\}, \quad \mathrm{emp}(h, S) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}_{\{h(\boldsymbol{x}_i) \neq y_i\}},$$

where $S = \{(\boldsymbol{x}_i, y_i)\}$ and $(\boldsymbol{x}_*, y_*)$ is drawn from the data distribution, independently of S. A compression scheme of size $d < n$ is a learning algorithm A for which we can find a mapping I from samples S to ordered d-subsets of $\{1, \ldots, n\}$ and another algorithm R mapping samples $\tilde{S}$ of size d to hypotheses $R(\tilde{S}) \in \mathcal{H}$, such that for every n-sample S we have that $A(S) = R(S_{I(S)})$, where $S_{I(S)} = \{(\boldsymbol{x}_i, y_i) \,|\, i \in I(S)\}$. This means that we can extract from each sample S a subsample $S_{I(S)}$ such that the algorithm essentially has to be trained on $S_{I(S)}$ only in order to produce the same result as on the full sample S. Once $I(S)$ is chosen, $S_{\setminus I(S)} = S \setminus S_{I(S)}$ is ignored. Thus, the data can be *compressed* before presenting it to the learner, and it seems intuitive that this characteristic can be exploited to prove generalisation error bounds for such schemes. A further characteristic that can be somewhat exploited is permutation-invariance of the scheme w.r.t. the data sample. If A is a d-compression scheme (with associated mappings I and R) and $0 \leq l \leq d$, we call A l-permutation-invariant if for every sample S, feeding R with a sample obtained from $S_{I(S)}$ by permuting the last l points leads to the same result, independent of the permutation. Special cases are $l = 0$ (we are not allowed to permute any points) and $l = d$ (we are allowed to permute all points). Herbrich [72], Sect. 5.2.1 provides further motivation and gives examples of compression schemes.

Fix $\delta \in (0,1)$, and choose $d \in \{1,\ldots,n\}$, $l \in \{0,\ldots,d\}$ *a priori* (this can be relaxed, as is shown below).

Theorem B.2 (PAC compression bound) *Suppose the algorithm A is, for samples S of size n, a l-permutation invariant d-compression scheme (with associated mappings I and R). Then, for any data distribution we have that the following bound holds, where the probability is over i.i.d. samples $S = \{(\boldsymbol{x}_i, y_i) \mid i = 1,\ldots,n\}$ of size n drawn from the data distribution:*

$$\Pr_S \left\{ \begin{array}{ll} \operatorname{gen}(A(S)) & \geq \operatorname{emp}\left(A(S), S_{\setminus I(S)}\right) \\ & +\varepsilon\left((n-d)\operatorname{emp}(A(S), S_{\setminus I(S)}), d, \tilde{\delta}\right) \end{array} \right\} \leq \delta, \tag{B.7}$$

where $\tilde{\delta} = \delta/(n-d+1)$ and $\varepsilon(q,d,\tilde{\delta}) = \max \mathrm{D}_{\mathrm{Ber}}^{-1}(q/(n-d),\rho)$ with

$$\rho = \frac{1}{n-d}\left(\log\left(\binom{n}{d}(d-l)!\right) + \log\tilde{\delta}^{-1}\right)$$

(recall the definition of $\mathrm{D}_{\mathrm{Ber}}^{-1}$ from Equation 3.1).

Note that the error on the remaining patterns $S_{\setminus I(S)}$, namely $\operatorname{emp}(A(S), S_{\setminus I(S)})$ is denoted by $\operatorname{emp}^{\setminus d}(S)$ in Section 3.5.4 and also below in this section.

As for the proof, fix $q \in \{0,\ldots,n-d\}$. Our first goal is to upper bound

$$\Pr_S\left\{\operatorname{emp}(A(S), S_{\setminus I(S)}) = \frac{q}{n-d},\ \ \operatorname{gen}(A(S)) \geq \frac{q}{n-d} + \varepsilon\right\}, \tag{B.8}$$

where $\varepsilon > 0$. Recall that $A(S) = R(S_{I(S)})$. We first apply a simple union bound argument, summing over all possible values of $I(S)$, in order to obtain the following upper bound on (B.8):

$$\sum_I \Pr_S\left\{\operatorname{emp}(R(S_I), S_{\setminus I}) = \frac{q}{n-d},\ \ \operatorname{gen}(R(S_I)) \geq \frac{q}{n-d} + \varepsilon\right\}. \tag{B.9}$$

We will determine the range and number of admissible I later. For now, fix I. We can upper bound the addend in (B.9) corresponding to I by

$$\mathrm{E}_{S_I}\left[\Pr_{S_{\setminus I}}\left\{\hat{p} \leq \frac{q}{n-d},\ \ p \geq \frac{q}{n-d} + \varepsilon \;\middle|\; S_I\right\}\right], \tag{B.10}$$

where we denote $p = \mathrm{gen}(R(S_I))$ and $\hat{p} = \mathrm{emp}(R(S_I), S_{\setminus I})$. Since $R(S_I)$ depends just on S_I, but not on $S_{\setminus I}$, we have that, conditioned on S_I, $(n-d)\,\hat{p}$ is $(n-d, p)$-binomial distributed. Therefore, we can use Chernoff's bound (Theorem A.2) to obtain

$$\Pr_{S_{\setminus I}}\left\{\hat{p} \leq \frac{q}{n-d},\ \ p \geq \frac{q}{n-d} + \varepsilon \;\middle|\; S_I\right\} \leq e^{-(n-d)\,\mathrm{D}_{\mathrm{Ber}}[q/(n-d)\,\|\,q/(n-d)+\varepsilon]}. \quad \text{(B.11)}$$

Since the r.h.s. in (B.11) does not depend on S_I, it is also an upper bound of (B.10). It is also independent of I, thus we can bound (B.9) by simply counting the number of admissible I, leading to potentially different $R(S_I)$. Since A is l-permutation invariant, we can permute the last l points in S_I without changing $R(S_I)$. Thus, there are $\binom{n}{d}(d-l)!$ distinguishable values for I. Altogether we have from (B.9)

$$\begin{aligned}&\Pr_S\left\{\mathrm{emp}(A(S), S_{\setminus I(S)}) = \frac{q}{n-d},\ \ \mathrm{gen}(A(S)) \geq \frac{q}{n-d} + \varepsilon\right\}\\ &\leq \binom{n}{d}(d-l)!\, e^{-(n-d)\,\mathrm{D}_{\mathrm{Ber}}[q/(n-d)\,\|\,q/(n-d)+\varepsilon]}.\end{aligned} \quad \text{(B.12)}$$

Equating the r.h.s. in (B.12) to $\tilde{\delta} > 0$ and solving for ε leads to the (unique) solution $\varepsilon = \varepsilon(q, k, \tilde{\delta})$. Therefore, we have for fixed q

$$\Pr_S\left\{\mathrm{emp}(A(S), S_{\setminus I(S)}) = \frac{q}{n-d},\ \ \mathrm{gen}(A(S)) \geq \frac{q}{n-d} + \varepsilon(q, d, \tilde{\delta})\right\} \leq \tilde{\delta},$$

where $\tilde{\delta} = \delta/(n-d+1)$. Noting that there are $n-d+1$ different values for q, we see that

$$\begin{aligned}&\Pr_S\left\{\begin{array}{ll}\mathrm{gen}(A(S)) & \geq \mathrm{emp}(A(S), S_{\setminus I(S)})\\ & \quad +\varepsilon\left((n-d)\,\mathrm{emp}(A(S), S_{\setminus I(S)}), d, \tilde{\delta}\right)\end{array}\right\}\\ &\leq \sum_{q=0}^{n-d}\Pr_S\left\{\begin{array}{l}\mathrm{emp}(A(S), S_{\setminus I(S)}) = \frac{q}{n-d},\\ \mathrm{gen}(A(S)) \geq \frac{q}{n-d} + \varepsilon(q, d, \tilde{\delta})\end{array}\right\} \leq \delta.\end{aligned}$$

This concludes the proof of Theorem B.2.

Note that we have proved Theorem B.2 under the assumption that d is fixed *a priori*. This can be relaxed, allowing for a dependence of d on the sample S, by using a further union bound argument. We end up with a theorem of exactly

the same form of Theorem B.2, however $\tilde{\delta}$ has to replaced by $\tilde{\delta} = 2\delta/((n+1)n)$. Here, we assume that l is a function of d, which is reasonable, however can be relaxed as well if desired.

Note that the main contribution to the gap bound value comes from the binomial coefficient (for not too small d) and is introduced by the crude union bound argument (summing over all I). This is necessary since we do not make any assumptions about the mapping $I(S)$. For a particular algorithm, it may be possible to come up with a more informed weighting than the uniform one (over the possible sets I), which might tighten the bound significantly. Also, note that in the special case that we attain $\mathrm{emp}^{\setminus d}(S) = 0$, we can solve for ε analytically, since $\mathrm{D}_{\mathrm{Ber}}[0 \,\|\, \varepsilon] = -\log(1-\varepsilon)$ (see Equation A.7), so that for $q = 0$, we have

$$\varepsilon(0, d, \tilde{\delta}) = 1 - \left[\binom{n}{d}(d-l)!\,\tilde{\delta}^{-1}\right]^{-1/(n-d)}.$$

We finally note that in many compression schemes, the index mapping I is partly randomised, i.e., is a function of the sample S and of random coin tosses. Since the latter are independent of S, this poses no problem for Theorem B.2, which holds for all possible outcomes of the coin tosses (although, of course, the bound value depends on these outcomes).

B.6.1 Examples of compression schemes

Herbrich [72], Sect. 5.2.1 gives a range of examples of compression schemes. It is shown there that the perceptron learning algorithm of Rosenblatt [154] is a 0-permutation invariant d-compression scheme, where d is the number of patterns the algorithm selects for an update of the weight vector. Note that, due to the perceptron convergence theorem, it is possible to upper bound d in terms of the margin of the data. Furthermore, the popular support vector machine can be seen as d-permutation invariant d-compression scheme, where d is the number of support vectors (this follows from the discussion in Section 2.1.6). Thus, our application of the compression bound in Section 3.5.4.1 is justified. Since the final SVM discriminant can make mistakes only on support vectors, we always

have $\mathrm{emp}^{\setminus d}(S) = 0$. Note that d cannot be fixed *a priori*, thus we have to use $\tilde{\delta} = 2\delta/((n+1)n)$ in Theorem B.2.

Furthermore, the IVM (see Section 4.4.1) is a compression scheme as well. This can be seen by noting that only patterns which are selected for inclusion into the active set, are ever used to update model parameters. In our experiments (see Section 3.5.3) we fixed d *a priori*, therefore Theorem B.2 applies in its original form if we set $l = d - d_{\mathrm{rand}}$. Note that in variants of the scheme d is chosen depending on the sample S, and in such cases we have to use the modified $\tilde{\delta}$.

Appendix C

Appendix for Chapter 4

In this chapter, we provide derivations and additional details in order to complete the presentation in Chapter 4.

C.1 Expectation Propagation

The EP algorithm [125] realises a general-purpose framework for approximating posterior beliefs by exponential family distributions. A generic introduction is given in Section C.1.1, serving to develop the Gaussian case in Section 4.3. Details of doing ADF or EP updates in this case are given in Section C.1.2.

C.1.1 Expectation Propagation for Exponential Families

In this section, we introduce the general expectation propagation (EP) algorithm [125], whose specialisation to GP models we require in Chapter 4 (see Section 4.3). Suppose we are given some statistical model with observables S and latent variables $\boldsymbol{u}$, with a prior distribution $P(\boldsymbol{u})$ from an exponential family $\mathcal{F}$. Some properties of exponential families which we require here, are given in Section A.4.1. The likelihood function $P(S|\boldsymbol{u})$ often factors in a particular way,

$$P(S|\boldsymbol{u}) = \prod_{i=1}^{n} t_i(\boldsymbol{u}),$$

for example in the case of i.i.d. data S or Bayesian networks. We refer to the $t_i(\boldsymbol{u})$ as *sites*. If the true posterior

$$P(\boldsymbol{u}|S) \propto P(\boldsymbol{u}) \prod_{i=1}^{n} t_i(\boldsymbol{u}) \tag{C.1}$$

is analytically intractable, we may approximate it by a distribution $Q(\boldsymbol{u})$ from $\mathcal{F}$:

$$Q(\boldsymbol{u}) = Q(\boldsymbol{u}|\boldsymbol{\theta}) = \exp\left(\boldsymbol{\theta}^T \boldsymbol{\varphi}(\boldsymbol{u}) + h(\boldsymbol{u}) - \Phi(\boldsymbol{\theta})\right), \ \boldsymbol{\theta} \in \Theta.$$

An often tractable way for choosing Q is to start from $Q(\boldsymbol{u}) = P(\boldsymbol{u})$ and incorporate the sites $t_i(\boldsymbol{u})$ one after the other, following some sequential ordering. Namely, in order to incorporate $t_i(\boldsymbol{u})$, first compute the true Bayesian update

$$\hat{P}(\boldsymbol{u}) = Z_i^{-1} Q(\boldsymbol{u}) t_i(\boldsymbol{u}), \quad Z_i = \mathrm{E}_{\boldsymbol{u}\sim Q}[t_i(\boldsymbol{u})],$$

then m-project it onto $\mathcal{F}$ (see Section A.4.2) to obtain the updated belief Q^{new}:

$$Q^{new}(\boldsymbol{u}) = \operatorname*{argmin}_{\tilde{Q}\in\mathcal{F}} \mathrm{D}\left[\hat{P}(\boldsymbol{u}) \,\|\, \tilde{Q}(\boldsymbol{u})\right].$$

Recall that m-projection is equivalent to *match moments* between $\hat{P}$ and Q^{new}. Namely, if $Q^{new}(\boldsymbol{u}) = Q^{new}(\boldsymbol{u}|\boldsymbol{\theta}^{new})$, and $\boldsymbol{\eta}$, $\boldsymbol{\eta}^{new}$ are the (dual) moment parameters for $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{new}$ (see Section A.4.1), then $\boldsymbol{\eta}^{new} = \mathrm{E}_{\hat{P}}[\boldsymbol{\varphi}(\boldsymbol{u})]$. We refer to this process as *inclusion* of site $t_i(\boldsymbol{u})$ into the belief Q. An inclusion is different from a true Bayesian update, since the full updated belief $\hat{P}$ is "collapsed" to Q^{new}, a member of $\mathcal{F}$, which allows inclusions to be chained. $\hat{P}$ lives in the tilted exponential family $\mathcal{F}_{t_i}$ induced by t_i (Definition A.6), and its moments can be computed via (A.12) which is often feasible (or amenable to numerical approximations) even though the moments of the full posterior $P(\boldsymbol{u}|S)$ remain intractable. This simple idea has been used extensively, for example in the context of Bayesian on-line learning [138] or switching linear dynamical systems [10, 25, 127], see [125] for more exhaustive references. It is known as *assumed density filtering (ADF)*. Nevertheless, each site may be included only once, and in the context of dynamical systems we are restricted to updates in one direction along the backbone chain (filtering), while bidirectional smoothing would maybe improve the approximation.

In [125], a new view on ADF is established which allows these shortcomings to be removed. The process of including $t_i(\boldsymbol{u})$ results in $Q(\boldsymbol{u})$ being replaced by $Q^{new}(\boldsymbol{u})$, which can also be seen as multiplying $Q(\boldsymbol{u})$ by the ratio $\tilde{t}_i(\boldsymbol{u}) \propto Q^{new}(\boldsymbol{u})/Q(\boldsymbol{u})$ and renormalising. This operation becomes particularly simple in the natural parameters: $\boldsymbol{\theta}^{new} = \boldsymbol{\theta} + (\boldsymbol{\theta}^{new} - \boldsymbol{\theta})$. The ratio $\tilde{t}_i(\boldsymbol{u})$ is a member of the unnormalised exponential family $\mathcal{F}^U$ associated with $\mathcal{F}$ (Definition A.7): it has a form very similar to Q and Q^{new}, but $\boldsymbol{\theta}^{new} - \boldsymbol{\theta}$ will not in general lie in the natural parameter space Θ of $\mathcal{F}$.[1] This view motivates *representing* Q as

$$Q(\boldsymbol{u}) \propto P(\boldsymbol{u}) \prod_{i=1}^{n} \tilde{t}_i(\boldsymbol{u}), \tag{C.2}$$

where the $\tilde{t}_i(\boldsymbol{u}) = \tilde{t}_i(\boldsymbol{u}|\boldsymbol{\theta}^{(i)}) \in \mathcal{F}^U$ are referred to as *site approximations*, and their natural parameters $\boldsymbol{\theta}^{(i)}$ as *site parameters*. If $\boldsymbol{\theta}^{(0)}$ denotes the parameters of $P(\boldsymbol{u})$, then

$$\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)} + \sum_{i=1}^{n} \boldsymbol{\theta}^{(i)}.$$

Note that we allow $\boldsymbol{\theta}^{(i)} = \mathbf{0}$, $\tilde{t}_i(\boldsymbol{u}) \equiv 1$, in fact in the beginning all site approximations are constant, leading to $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)}$, i.e. $Q(\boldsymbol{u}) = P(\boldsymbol{u})$. An *ADF update (inclusion)* w.r.t. site t_i can now be seen as follows (note that $\tilde{t}_i \equiv 1$):

Definition C.1 (ADF Update, Inclusion)

1. *Compute moments of* $\hat{P}(\boldsymbol{u}) \propto Q(\boldsymbol{u})t_i(\boldsymbol{u})$ *and pick* $Q^{new}(\boldsymbol{u}) \in \mathcal{F}$ *with these moments.*

2. *In order to replace* $Q(\boldsymbol{u})$ *by* $Q^{new}(\boldsymbol{u})$*, we replace* $\tilde{t}_i(\boldsymbol{u}) \equiv 1$ *by* $\tilde{t}_i^{new}(\boldsymbol{u}) \propto Q^{new}(\boldsymbol{u})/Q(\boldsymbol{u})$.

From this viewpoint, it becomes clear how ADF can be generalised to a full-fledged iterative approximation scheme, allowing for multiple iterations over the sites. An *EP update (inclusion-deletion)* w.r.t. site t_i works as follows:

[1] For example, if $\mathcal{F}$ is the family of Gaussians, then $\tilde{t}_i$ may correspond to a "Gaussian with negative variance".

Definition C.2 (EP Update, Inclusion-Deletion)

1. *Delete the site approximation* $\tilde{t}_i(\boldsymbol{u})$ *from* $Q(\boldsymbol{u})$ *by renormalising* $Q(\boldsymbol{u})/\tilde{t}_i(\boldsymbol{u})$*, obtaining*
$$Q^{\backslash i}(\boldsymbol{u}) \propto P(\boldsymbol{u}) \prod_{j \neq i} \tilde{t}_j(\boldsymbol{u}).$$
In natural parameters: $\boldsymbol{\theta}^{\backslash i} = \boldsymbol{\theta} - \boldsymbol{\theta}^{(i)}$.

2. *Let* $\hat{P}(\boldsymbol{u}) = Z_i^{-1} t_i(\boldsymbol{u}) Q^{\backslash i}(\boldsymbol{u})$ *and compute*
$$\boldsymbol{\eta}^{new} = \mathrm{E}_{\hat{P}}[\boldsymbol{\varphi}(\boldsymbol{x})] = \nabla_{\boldsymbol{\theta}^{\backslash i}} \log Z_i + \boldsymbol{\eta}^{\backslash i}, \quad Z_i = \mathrm{E}_{\boldsymbol{\theta}^{\backslash i}}[t_i(\boldsymbol{u})],$$
and pick $Q^{new} \in \mathcal{F}$ *with these moments.*

3. *Replace* $\tilde{t}_i$ *by* $\tilde{t}_i^{new}(\boldsymbol{u}) \propto Q^{new}(\boldsymbol{u})/Q^{\backslash i}(\boldsymbol{u})$.
In natural parameters: $\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{new} - \boldsymbol{\theta}^{\backslash i}$.

In line with [139], we will refer to $Q^{\backslash i}$ as *cavity distribution*.

On networks with discrete nodes or Gaussian Markov random fields, EP can be seen as generalisation of loopy belief propagation, allowing for a more flexible choice of approximating structure and distribution family (see [125] and [203], Chap. 6). The algorithm does not always converge, but if it does, the fixed point must be a saddle points of an approximation to the free energy which is a generalisation of the Bethe free energy [123, 74, 203]. Double-loop concave-convex algorithms can be applied in order to ensure convergence [74]. Problems of convergence can sometimes be overcome by using "damped" updates: instead of $\boldsymbol{\theta} \to \boldsymbol{\theta}^{new}$, we update $\boldsymbol{\theta}$ to a convex combination of $\boldsymbol{\theta}$ and $\boldsymbol{\theta}^{new}$. Also, updates which lead to $\boldsymbol{\theta}^{new}$ outside of Θ (or very close to its boundary) should be rejected. In practice, it is important to address the issue of numerical stability: the conversion between natural and moment parameters is typically not a stable operation.[2] If possible, an implementation should remain in the moment parameters entirely and fold conversions with update operations into a single mapping which can then

[2]For example, for the Gaussian family we have to invert a matrix.

be stabilised. This is of course less generic than the presentation above, and it is also not clear how to do damped updates (which are convex combinations in the natural parameters) in this way.

It is important to note that EP is not simply a local approximation of the sites $t_i(\boldsymbol{u})$ by corresponding $\tilde{t}_i(\boldsymbol{u})$, but a global fit by $Q \in \mathcal{F}$ to the distribution obtained by replacing $\tilde{t}_i(\boldsymbol{u})$ by $t_i(\boldsymbol{u})$ in the current belief $Q(\boldsymbol{u})$. In fact, the sites may not even be continuous functions of $\boldsymbol{u}$. EP has been applied for approximate inference in two very different regimes: sparsely connected Bayesian or Markov networks and models with fully connected Gaussian prior $P(\boldsymbol{u})$. In the former regime, every single $t_i(\boldsymbol{u})$ depends on a small number of components of $\boldsymbol{u}$ only, e.g. in the Markov network case on small cliques of the underlying graph. By choosing a special structure of the approximating distribution Q, based on a tractable subgraph of the decomposable extension of the model graph, *and* requiring that the prior $P(\boldsymbol{u})$ follows this structure, one can run EP as a message-passing scheme, updating the parameters of Q and certain small extensions thereof. This notion is developed and generalised in [203], Chap. 6 (for discrete variables), and in [123] (does not address the issue of how to represent Q). In the second regime, which we are interested here, $\mathcal{F}$ is the family of Gaussians, and $P(\boldsymbol{u})$ is typically densely connected, while the likelihood factors $t_i(\boldsymbol{u})$ are local again. The special case for a completely factorised likelihood $P(S|\boldsymbol{u})$ has been given in [41]. Manipulations on Gaussians will in general be $O(n^3)$, which is the cost for converting between natural and moment parameters. Nevertheless, a *locality property* of EP can be used here as well in order to simplify EP updates and the representation of the belief Q. Let $\boldsymbol{u} = (\boldsymbol{u}_J^T, \boldsymbol{u}_{\setminus J}^T)^T$, and suppose that site $t_i(\boldsymbol{u})$ depends on $\boldsymbol{u}_J$ only, i.e. $t_i(\boldsymbol{u}) = t_i(\boldsymbol{u}_J)$. Then, $\hat{P}(\boldsymbol{u}) = \hat{P}(\boldsymbol{u}_J)Q^{\setminus i}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J)$ and

$$\begin{aligned}\mathrm{D}[\hat{P}(\boldsymbol{u}) \,\|\, Q^{new}(\boldsymbol{u})] = \mathrm{D}[\hat{P}(\boldsymbol{u}_J) \,\|\, Q^{new}(\boldsymbol{u}_J)] \\ + \mathrm{E}_{\boldsymbol{u}_J \sim \hat{P}}\left[\mathrm{D}[Q^{\setminus i}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J) \,\|\, Q^{new}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J)]\right].\end{aligned}$$

In order to minimise this expression, we set $Q^{new}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J) = Q^{\setminus i}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J)$ and match moments between the marginals $\hat{P}(\boldsymbol{u}_J)$ and $Q^{new}(\boldsymbol{u}_J)$. It follows that $\tilde{t}_i(\boldsymbol{u}) = \tilde{t}_i(\boldsymbol{u}_J)$, i.e. the site approximations inherit the locality of the corresponding sites and can be parameterised economically, in the sense that many

of the components in $\boldsymbol{\theta}^{(i)}$ can be clamped to zero. Furthermore, since $Q(\boldsymbol{u}) \propto Q^{\setminus i}(\boldsymbol{u})\tilde{t}_i(\boldsymbol{u}_J)$, we see that $Q(\boldsymbol{u}_J) \propto Q^{\setminus i}(\boldsymbol{u}_J)\tilde{t}_i(\boldsymbol{u}_J)$, so that in order to update the site approximation $\tilde{t}_i$, we only need to access the marginal $Q(\boldsymbol{u}_J)$: although EP is a global approximation, its updates will be local. Note however that a change of $\tilde{t}_i(\boldsymbol{u}_J)$ in general affects all marginals of $Q(\boldsymbol{u})$, due to the densely connected prior. We will make use of the locality property below, furthermore of the following lemma.

Lemma C.1 *If $Q(\boldsymbol{u})$, $\tilde{Q}(\boldsymbol{u})$ are Gaussians and $Q(\boldsymbol{u}_J|\boldsymbol{u}_{\setminus J}) = \tilde{Q}(\boldsymbol{u}_J|\boldsymbol{u}_{\setminus J})$, where $\boldsymbol{u} = (\boldsymbol{u}_J^T, \boldsymbol{u}_{\setminus J}^T)^T$, then*

$$\mathrm{H}[Q(\boldsymbol{u})] - \mathrm{H}[\tilde{Q}(\boldsymbol{u})] = \mathrm{H}[Q(\boldsymbol{u}_J)] - \mathrm{H}[\tilde{Q}(\boldsymbol{u}_J)].$$

and

$$\mathrm{D}[Q(\boldsymbol{u}) \,\|\, \tilde{Q}(\boldsymbol{u})] = \mathrm{D}[Q(\boldsymbol{u}_J) \,\|\, \tilde{Q}(\boldsymbol{u}_J)].$$

As for the proof, note that

$$\mathrm{H}[Q(\boldsymbol{u})] - \mathrm{H}[Q(\boldsymbol{u}_J)] = \mathrm{E}_{\boldsymbol{u}_J \sim Q}\left[\mathrm{H}[Q(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J)]\right] = \mathrm{E}_{\boldsymbol{u}_J \sim Q}\left[\mathrm{H}[\tilde{Q}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J)]\right].$$

The entropy of a Gaussian distribution depends on its covariance matrix only (Lemma A.13), and the covariance matrix of $\tilde{Q}(\boldsymbol{u}_{\setminus J}|\boldsymbol{u}_J)$ does not depend on $\boldsymbol{u}_J$, again a special property of Gaussian distributions (Lemma A.9). Therefore, we can take the expectation just as well over $\boldsymbol{u}_J \sim \tilde{Q}$, and this concludes the proof of the first part of the Lemma. The second part is straightforward.

C.1.2 ADF Update for some Noise Models

Here, we show how to do ADF updates (Definition C.1) or EP updates (Definition C.2) for some noise models used in this thesis, in the case of Gaussian beliefs. In general, let $t_i(\boldsymbol{u}) = t_i(\boldsymbol{u}_J)$ for $J \subset \{1, \ldots, n\}$. In order to update the site approximation $\tilde{t}_i(\boldsymbol{u}_J) = N^U(\boldsymbol{u}_J|\boldsymbol{b}_i, \boldsymbol{\Pi}_i)$, we require the marginal $Q(\boldsymbol{u}_J) = N(\boldsymbol{h}_J, \boldsymbol{A}_J)$. First, we compute $Q^{\setminus i}(\boldsymbol{u}_J) = N(\boldsymbol{h}_J^{\setminus i}, \boldsymbol{\Lambda})$ by converting to natural parameters, subtracting off $\boldsymbol{b}_i$, $\boldsymbol{\Pi}_i$ and converting back. This results in

$$\boldsymbol{\Lambda} = \left(\boldsymbol{A}_J^{-1} - \boldsymbol{\Pi}_i\right)^{-1} = \boldsymbol{A}_J\left(\boldsymbol{I} - \boldsymbol{\Pi}_i\boldsymbol{A}_J\right)^{-1}, \quad \boldsymbol{h}_J^{\setminus i} = \boldsymbol{h}_J + \boldsymbol{\Lambda}\left(\boldsymbol{\Pi}_i\boldsymbol{h}_J - \boldsymbol{b}_i\right).$$

For an ADF update, $\boldsymbol{\Lambda} = \boldsymbol{A}_J$ and $\boldsymbol{h}_J^{\backslash i} = \boldsymbol{h}_J$, because the site parameters $\boldsymbol{b}_i, \boldsymbol{\Pi}_i$ have been zero. Next, we need the moments of the tilted Gaussian $\hat{P}(\boldsymbol{u}_J) \propto t_i(\boldsymbol{u}_J) Q^{\backslash i}(\boldsymbol{u}_J)$, which can be computed using (A.14). Let $Z_i = \mathrm{E}_{Q^{\backslash i}}[t_i(\boldsymbol{u}_J)]$. In the Gaussian context, it is more direct to compute derivatives of $\log Z_i$ w.r.t. the mean $\boldsymbol{h}_J^{\backslash i}$, but since $\boldsymbol{h}_J^{\backslash i} = \boldsymbol{\Lambda}\boldsymbol{r}$, $\boldsymbol{r}$ the vector natural parameter of $Q^{\backslash i}$, these are readily converted. Let

$$\boldsymbol{\alpha}_i = \nabla_{\boldsymbol{h}_J^{\backslash i}} \log Z_i, \quad \boldsymbol{\nu}_i = -\nabla\nabla_{\boldsymbol{h}_J^{\backslash i}} \log Z_i.$$

Then,

$$\boldsymbol{h}_J^{new} = \boldsymbol{h}_J^{\backslash i} + \nabla_{\boldsymbol{r}} \log Z_i = \boldsymbol{h}_J^{\backslash i} + \boldsymbol{\Lambda}\boldsymbol{\alpha}_i, \quad \boldsymbol{A}_J^{new} = (\boldsymbol{I} - \boldsymbol{\Lambda}\boldsymbol{\nu}_i)\,\boldsymbol{\Lambda}.$$

To compute the new site parameters, we convert these into natural parameters and subtract off the natural parameters of $Q^{\backslash i}$:

$$\boldsymbol{\Pi}_i^{new} = \boldsymbol{\nu}_i\,(\boldsymbol{I} - \boldsymbol{\Lambda}\boldsymbol{\nu}_i)^{-1}, \quad \boldsymbol{b}_i^{new} = \boldsymbol{\Pi}_i^{new}\left(\boldsymbol{h}_J^{\backslash i} + \boldsymbol{\nu}_i^{-1}\boldsymbol{\alpha}_i\right).$$

Note that if the likelihood $P(S|\boldsymbol{u})$ of the model factorises completely over $\boldsymbol{u}$, then $|J| = 1$ and the matrices and vectors simplify to scalars.

In case of binary classification and a probit noise model $P(y_i|u_i) = \Phi(y_i(u_i + b))$, Φ the c.d.f. of $N(0,1)$, i.e. $J = \{i\}$, we have

$$Z_i = \int \Phi(y_i(u_i + b)) Q^{\backslash i}(u_i)\, du_i = \Phi\left(\frac{y_i(h_i^{\backslash i} + b)}{\sqrt{1 + \lambda_i}}\right)$$

(see Equation 4.9). Then,

$$z_i = \frac{y_i(h_i^{\backslash i} + b)}{\sqrt{1 + \lambda_i}}, \quad \alpha_i = \frac{y_i N(z_i|0,1)}{\Phi(z_i)\sqrt{1 + \lambda_i}}, \quad \nu_i = \alpha_i\left(\alpha_i + \frac{h_i^{\backslash i} + b}{1 + \lambda_i}\right).$$

In practice, the direct evaluation of $N(z_i|0,1)/\Phi(z_i)$ is instable for $z_i \ll 0$, and our implementation evaluates a linear lower bound instead.

For general sites $t_i(\boldsymbol{u}_J)$, the Gaussian integrals for $\log Z_i$ and its derivatives are not analytically tractable. If $|J| = 1$, the method of choice for approximating Gaussian integrals is *Gaussian quadrature* (of the Gauss-Hermite type, e.g. [145], Sect. 4.5). This method delivers approximations to integrals

$$\int f(x) e^{-x^2}\, dx,$$

requiring the evaluation of $f(x)$ at a few chosen points only. In a nutshell, Gaussian quadrature constructs the unique set of polynomials orthonormal w.r.t. the inner product

$$(f, g) = \int f(x)g(x)e^{-x^2}\,dx,$$

uses the zeros of the degree N member as abscissas and adjusts the weights s.t. the rule is exact for $f(x)$ being a polynomial of order $\leq N-1$. It turns out that the corresponding rule is exact for all polynomials of order $\leq 2N-1$, owing to the d.o.f.'s for the free choice of abscissas. Thus, its general accuracy hinges on $f(x)$ being smooth s.t. it can be well-approximated by a polynomial in the region where $|f(x)e^{-x^2}|$ is significantly different from 0. If this does not hold true for $f(x)$ (in our case a suitable linear transformation of $t_i(u_i)$) it may be necessary to split the integral into parts or deviate from the standard quadrature rules. For $|J| > 1$, Gaussian quadrature is less attractive because the number of grid points required grows exponentially in $|J|$. Relevant literature for this problem is reviewed in [102]. In general, if $|J|$ is large one may have to resort to Monte Carlo (MC) techniques but their accuracy for rather small $|J|$ is poor compared to quadrature techniques. The method of *exact monomials* [120, 47] is suitable in our context exploiting the symmetry under orthogonal transformation of the weight function $N(\mathbf{0}, \boldsymbol{I})$.

C.2 Likelihood Approximations. Selection Criteria

In this section, we provide additional details concerning likelihood approximations and greedy selection criteria for sparse GP approximations.

C.2.1 Optimal Sparse Likelihood Approximations

Recall the notations used in Section 4.2.1. In this section, we give a proof for Lemma 4.1. For the first part (4.2), note that $Q(S|\boldsymbol{u}_I)$ is only determined up to a constant factor. Let us remove this indeterminacy by requiring $\mathrm{E}_{\boldsymbol{u}_I \sim P}[Q(S|\boldsymbol{u}_I)] = 1$, i.e. $Q(\boldsymbol{u}|S) = Q(S|\boldsymbol{u}_I)P(\boldsymbol{u})$. The problem of minimising (4.2) subject to this

constraint has the Lagrangian

$$\mathcal{L} = -\mathrm{E}_{P(\boldsymbol{u}_I|S)}\left[\log Q(S|\boldsymbol{u}_I)\right] + \lambda \mathrm{E}_{P(\boldsymbol{u}_I)}\left[Q(S|\boldsymbol{u}_I)\right],$$

leading to $Q(S|\boldsymbol{u}_I) = \lambda^{-1} P(\boldsymbol{u}_I)^{-1} P(\boldsymbol{u}_I|S)$. Finally,

$$P(\boldsymbol{u}_I|S) \propto \int P(S|\boldsymbol{u})P(\boldsymbol{u})\, d\boldsymbol{u}_{\setminus I} = P(\boldsymbol{u}_I)\mathrm{E}_P[P(S|\boldsymbol{u}) \,|\, \boldsymbol{u}_I],$$

so that $Q(S|\boldsymbol{u}_I) \propto \mathrm{E}_P[P(S|\boldsymbol{u}) \,|\, \boldsymbol{u}_I]$. For the second part (4.3), we note that the relative entropy term can be rewritten (up to a constant) as

$$\mathrm{E}_{Q(\boldsymbol{u}_I|S)}\left[\log \frac{Q(\boldsymbol{u}_I|S)}{\exp\left(\mathrm{E}_P[\log P(S|\boldsymbol{u}) \,|\, \boldsymbol{u}_I]\right) P(\boldsymbol{u}_I)}\right],$$

where we used the fact that the prior terms are the same and cancel out. This is minimised by making sure that numerator and denominator are proportional.[3] Note that this is essentially the same argument as the one used to derive free-form optimal smoothing distributions in variational approximations (see Section A.6.3). Now, suppose that $P(S|\boldsymbol{u}) = N^U(\boldsymbol{u}|\boldsymbol{b}, \boldsymbol{\Pi})$ and denote by $\boldsymbol{\mu} = \mathrm{E}_P[\boldsymbol{u}|\boldsymbol{u}_I]$ the conditional mean of $\boldsymbol{u}$ given $\boldsymbol{u}_I$ under the prior P. Then,

$$\mathrm{E}_P[\log P(S|\boldsymbol{u}) \,|\, \boldsymbol{u}_I] \propto -\frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\Pi}\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{b},$$

simply $\log P(S|\boldsymbol{u})$ is quadratic in $\boldsymbol{u}$ and for a conditional from a joint Gaussian distribution, the conditional covariance matrix does not depend on the variable we condition on (Lemma A.9). Therefore, the optimal likelihood approximation in the sense of statement (4.3) in the Gaussian case is

$$Q(S|\boldsymbol{u}_I) \propto N^U\left(\mathrm{E}_P[\boldsymbol{u}|\boldsymbol{u}_I] \,\middle|\, \boldsymbol{b}, \boldsymbol{\Pi}\right). \tag{C.3}$$

The important fact in the context of Chapter 4 is that the conditional mean $\boldsymbol{\mu}$ can be computed by inverting the *marginal* covariance of $P(\boldsymbol{u}_I)$ only. Unfortunately, in the case of the first statement (4.2), the form of the optimal likelihood approximation requires the computation of a d-by-d block from the *inverse* covariance matrix of $P(\boldsymbol{u})$, which is not feasible for sparse approximations (see Lemma A.1).

[3] Recall that the relative entropy (Definition A.4) is non-negative, and 0 iff its arguments are identical distributions.

C.2.2 Relaxed Likelihood Approximations

Recall the discussion in Section 4.2.1. From (4.1) it is clear that in general the $O(n^2)$ computation of $\boldsymbol{K}$ can be avoided only if $\boldsymbol{\Pi} = \boldsymbol{I}_{\cdot,I}\boldsymbol{E}\boldsymbol{I}_{I,\cdot}$ for symmetric nonsingular $\boldsymbol{E} \in \mathbb{R}^{d,d}$. However, if $\mu(\boldsymbol{x}_*)$ is allowed to depend on all of $\boldsymbol{k}_*$ the restriction on $\boldsymbol{r}$ can be dropped. Let $\boldsymbol{E} = \boldsymbol{F}\boldsymbol{F}'$. Then, the prediction vector

$$\boldsymbol{\xi} = \boldsymbol{r} - \boldsymbol{I}_{\cdot,I}\boldsymbol{F}\left(\boldsymbol{I} - \boldsymbol{F}'\boldsymbol{K}_I\boldsymbol{F}\right)^{-1}\boldsymbol{F}'\boldsymbol{K}_{I,\cdot}\boldsymbol{r}$$

can be computed within our resource constraints (which allow for the evaluation of $\boldsymbol{K}_{I,\cdot}$).

Note that our parameterisation translates directly into the form for the likelihood approximation which is

$$Q(S|\boldsymbol{u}) = N^U(\boldsymbol{u}|\boldsymbol{r}, \boldsymbol{\Pi}).$$

This can be brought into a centred form $N^{UC}(\boldsymbol{\mu}, \boldsymbol{\Pi})$ iff there exists a $\boldsymbol{\mu}$ s.t. $\boldsymbol{\Pi}\boldsymbol{\mu} = \boldsymbol{r}$, and in this case $\boldsymbol{r}_{\setminus I} = \boldsymbol{0}$. However, the existence of such a $\boldsymbol{\mu}$ is not a strict requirement for a likelihood approximation. For example, consider the MRED view of large margin classifiers (Section 2.1.6) whose "log likelihood" is linear in $\boldsymbol{u}$ (although there are additional constrained margin variables, adding nonlinear dependencies). With $\boldsymbol{r}_{\setminus I} \neq \boldsymbol{0}$ the likelihood approximation depends on $\boldsymbol{u}_{\setminus I}$ in a log-linear way:

$$Q(S|\boldsymbol{u}) = N^U(\boldsymbol{u}_I|\boldsymbol{r}_I, \boldsymbol{E}) \exp\left(\boldsymbol{r}_{\setminus I}^T\boldsymbol{u}_{\setminus I}\right).$$

Whether these additional d.o.f.'s can be used to obtain more accurate approximations to the true posterior remains open here.

C.2.3 Cheap versus Expensive Selection Criteria

In Section 4.2.2 we noted that for the task of subset selection, cheap criteria should generally be preferred over expensive ones. In this context, "expensive" means that a score evaluation is as costly as the actual inclusion, i.e. computation of Q_i^{new}. But if time was not an issue, would a sensible expensive criterion always

be preferable over a cheap one? Here, we give a simple example that this need not be the case. Within given resource constraints, criteria should be chosen according to characteristics and requirements of the task.

Given the symmetric positive definite linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, we would like to find a sparse approximate solution $\tilde{\boldsymbol{x}} = \boldsymbol{I}_{\cdot,I}\boldsymbol{\beta}$, where I has a given size d. Here, $\boldsymbol{A}$ is not available in memory and has to be computed on demand. Let $\boldsymbol{x}_* = \boldsymbol{A}^{-1}\boldsymbol{b}$. Minimising $\|\tilde{\boldsymbol{x}} - \boldsymbol{x}_*\|$ directly is not feasible incrementally, but we can try to minimise $\phi_{\mathrm{CG}}(\boldsymbol{x}) = (1/2)\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}$ or $\phi_{\mathrm{LS}}(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|^2$ by greedy forward selection: in each iteration, every $i \notin I$ is scored by the maximum reduction in the criterion achievable if component i of $\tilde{\boldsymbol{x}}$ is allowed to be non-zero. If $|I| = d$, this score is $O(d)$ for ϕ_{CG}, but requires the evaluation of the column $\boldsymbol{A}_{\cdot,i}$ for ϕ_{LS}: ϕ_{CG} is cheap, ϕ_{LS} expensive. But if $\|\boldsymbol{u}\|_{\boldsymbol{B}}^2 = \boldsymbol{u}^T\boldsymbol{B}\boldsymbol{u}$, then $\phi_{\mathrm{CG}}(\boldsymbol{x})$ is equal up to constants to $\|\boldsymbol{x} - \boldsymbol{x}_*\|_{\boldsymbol{A}}^2$, while $\phi_{\mathrm{LS}}(\boldsymbol{x})$ corresponds to $\|\boldsymbol{x} - \boldsymbol{x}_*\|_{\boldsymbol{A}^2}^2$. If our goal is a good approximation w.r.t. $\|\boldsymbol{x} - \boldsymbol{x}_*\|$, then ϕ_{CG} should actually be preferred as a criterion in this case, because even if ϕ_{CG} over-penalises errors along eigendirections with high $\boldsymbol{A}$ eigenvalues, ϕ_{LS} does so much stronger. In this particular case, the task requirements actually lead us to prefer the cheaper criterion, even if we could afford to use the expensive one.

C.3 Derivations for the Informative Vector Machine

In this section, we collect details concerning the IVM approximation, as introduced in Section 4.4.1.

C.3.1 Update of the Representation

Recall the notation and definitions used in Section 4.4.1, and also recall that in order to decide which point to include next into the active set I, we *score* all remaining points (or a significant fraction thereof). In order to compute a score for point $j \notin I$, we need to know the corresponding marginal $Q(u_j)$, i.e. the corresponding elements of $\boldsymbol{h}$ and $\operatorname{diag}\boldsymbol{A}$. Here, we describe an efficient and stable scheme for maintaining this information. We will first assume that all marginals

have to be kept up-to-date between inclusions. Further below, we relax this requirement to end up with a *randomised* greedy scheme.

Recall that $\boldsymbol{A} = (\boldsymbol{K}^{-1} + \boldsymbol{\Pi})^{-1}$. Using the Woodbury formula (Lemma A.2) we can write

$$\boldsymbol{A} = \boldsymbol{K} - \boldsymbol{K}_{I,\cdot}^T \boldsymbol{\Pi}_I^{1/2} \boldsymbol{B}^{-1} \boldsymbol{\Pi}_I^{1/2} \boldsymbol{K}_{I,\cdot}, \quad \boldsymbol{B} = \boldsymbol{I} + \boldsymbol{\Pi}_I^{1/2} \boldsymbol{K}_I \boldsymbol{\Pi}_I^{1/2}. \tag{C.4}$$

We maintain a Cholesky decomposition $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T$ (see Section A.2.2) with which $\boldsymbol{A} = \boldsymbol{K} - \boldsymbol{M}^T\boldsymbol{M}$, where

$$\boldsymbol{M} = \boldsymbol{L}^{-1} \boldsymbol{\Pi}_I^{1/2} \boldsymbol{K}_{I,\cdot} \tag{C.5}$$

is a matrix we have to maintain in memory. Note that $\boldsymbol{h} = \boldsymbol{M}^T\boldsymbol{\beta}$, using (4.7). In short, knowledge of the marginal $Q(u_i)$ hinges on column $\boldsymbol{M}_{\cdot,i}$ of $\boldsymbol{M}$. Suppose a new point i is to be included into I, and its site parameters b_i and π_i have already been computed. We have to update $\boldsymbol{L} \to \boldsymbol{L}' \in \mathbb{R}^{d+1,d+1}$ and $\boldsymbol{M} \to \boldsymbol{M}' \in \mathbb{R}^{d+1,n}$. First, $\boldsymbol{L}'_{1\dots d,1\dots d} = \boldsymbol{L}$ and $\boldsymbol{l}_i = \boldsymbol{L}'_{1\dots d,d+1} = \boldsymbol{L}^{-1}\pi_i^{1/2}\boldsymbol{\Pi}_I^{1/2}\boldsymbol{K}_{I,i} = \pi_i^{1/2}\boldsymbol{M}_{\cdot,i}$, furthermore $l_i = \boldsymbol{L}'_{d+1,d+1} = (1 + \pi_i\boldsymbol{K}_{i,i} - \boldsymbol{l}_i^T\boldsymbol{l}_i)^{1/2} = (1 + \pi_i\sigma^2(\boldsymbol{x}_i))^{1/2}$. Next, $(\boldsymbol{M}')_{1\dots d,\cdot} = \boldsymbol{M}$ and $\boldsymbol{\mu} = (\boldsymbol{M}')_{d+1,\cdot}^T = l_i^{-1}(\pi_i^{1/2}\boldsymbol{K}_{\cdot,i} - \boldsymbol{M}^T\boldsymbol{l}_i)$, furthermore $\operatorname{diag}\boldsymbol{A}^{new} = \operatorname{diag}\boldsymbol{A} - (\mu_j^2)_j$. For the update of $\boldsymbol{h}$, note that $\boldsymbol{h}^{new} = \boldsymbol{h} + \beta^{new}\boldsymbol{\mu}$, where β^{new} is the new component of $\boldsymbol{\beta}^{new}$. We can compute this directly, by observing that $\boldsymbol{h}^{new} = \boldsymbol{h} + \eta\boldsymbol{a}_i$, since $\boldsymbol{A}^{new} - \boldsymbol{A} \propto \boldsymbol{a}_i\boldsymbol{a}_i^T$ and $\boldsymbol{b}^{new} - \boldsymbol{b} \propto \boldsymbol{\delta}_i$. The factor η can be determined by solving $h_i^{new} = h_i + \eta a_{i,i}$, resulting in $\eta = \alpha_i$ (see Section C.1.2 and recall that we are doing ADF updates only). It is also easy to see that $\boldsymbol{a}_i = l_i\pi_i^{-1/2}\boldsymbol{\mu}$, thus $\boldsymbol{h}^{new} = \boldsymbol{h} + \alpha_i l_i \pi_i^{-1/2}\boldsymbol{\mu}$ and $\beta^{new} = \alpha_i l_i \pi_i^{-1/2}$. The storage requirements are dominated by $\boldsymbol{M}$, the running time requirements by the computation of the new row $\boldsymbol{\mu}$ of $\boldsymbol{M}$, which is $O(n\,d)$ and the evaluation of column i of the kernel matrix.

C.3.1.1 Randomised Greedy Selection

Recall from Section 4.4.1 that for randomised greedy selection (RGS), we have to keep only a subset of the marginals up-to-date, corresponding to a selection index J. The speed-up comes from the fact that not all columns of $\boldsymbol{M}$ have to be kept up-to-date at any time, we can use the following delayed updating

scheme. To this end, we maintain an integer vector $\boldsymbol{g} \in \mathbb{N}^n$. For any $i \notin I$, g_i is the iteration number of the last recent inclusion for which column i of $\boldsymbol{M}$ was up-to-date. Denote by $\boldsymbol{L}^{(d)} \in \mathbb{R}^{d,d}$, $\boldsymbol{M}^{(d)} \in \mathbb{R}^{d,n}$ the matrices $\boldsymbol{L}$, $\boldsymbol{M}$ after the inclusion of d points. For each $i \notin I$, we know that $\boldsymbol{M}_{1\dots g_i,i} = \boldsymbol{M}^{(g_i)}_{1\dots g_i,i}$, where $\boldsymbol{M}$ is the current buffer matrix. We call this vector $\in \mathbb{R}^{g_i}$ the *stub* for i. Now suppose we have included d points already and are about to include a further point, after which the selection index will be modified to J. In order to update the components of $\operatorname{diag} \boldsymbol{A}$ and $\boldsymbol{h}$ corresponding to J, we have to go through all $j \in J$ and make sure that all stubs corresponding to the new J have full size $d+1$. Now, it is central to note that the back-substitution with $\boldsymbol{L}^{(d+1)}$ required to do this is naturally broken up into steps $g = 1, \dots, d+1$, s.t. step g uses the g-th row of $\boldsymbol{L}^{(d+1)}$ only and produces element g of the stub. Thus, we can save the first g_j steps of this procedure and directly start with the stub of j, then do steps $g = g_j + 1, \dots, d+1$ as described above, finally set $g_j = d+1$. Note that in order for this randomised scheme to be efficient, J not only has to be significantly smaller than $\{1, \dots, n\} \setminus I$, but also has to be exhibit some "inertia" over time.

If n and d are not too large, a simple implementation would just use $O(n\,d)$ memory. As long as this is possible without exhausting the system's RAM, this is the most efficient option in terms of running time. If not, a cache structure should be used. Our present implementation uses a simple structure consisting of two buffers. The smaller one keeps $M_{\cdot,J}$ (size $d \times |J|$), the larger one has $n - |J|$ columns of size at most d_{lim}, where d_{lim} should be chosen as large as possible within resource constraints. Whenever $j_1 \in J$ is to be replaced by j_2, we exchange their stubs in the corresponding buffers. This may lead to the stub of j_1 being trimmed to size d_{lim}. In this case, we chop the first $g_{j_1} - d_{lim}$ elements, because these can be re-computed faster later, should j_1 return into J at some later time. This scheme is fairly simple to implement, but may need to be modified if very large n are to be used. In this case, the process of selecting new patterns for inclusion into J should not be uniform, but prefer patterns which have been in J before. Also, a retain fraction τ which grows with d should be considered. One can then implement a cache hierarchy which leads to less trimming and re-

computation. A general problem in practice with RGS is that the dominating computational steps which scale with n for full selection, have to be done one-by-one. For full greedy selection, the large back-substitutions can be done en-block, having the inner-most loops scale with n or even making use of optimised software like BLAS,[4] while for randomised selection this is not possible. As much as one would like to ignore such issues, they become painfully important when dealing with very large n, but this is not within the scope of this thesis.

C.3.2 Exchange Moves

In this section, we derive update equations for exchange moves which at the same time remove $j \in I$ from the active set and include $i \notin I$, thus leaving the total size constant. Exchange candidates (j, i) can be scored in much the same way as inclusion candidates.

Both the move itself and the score computation require knowledge of the marginal $Q(u_i, u_j)$. By the locality property of EP (Lemma C.1), $Q^{new}(\boldsymbol{u}^{\backslash i}|u_i) = Q^{\backslash j}(\boldsymbol{u}^{\backslash i}|u_i)$ and $\tilde{t}_i^{new}(u_i) \propto Q^{new}(u_i)/Q^{\backslash j}(u_i)$. Although here $Q^{\backslash j}(\boldsymbol{u}^{\backslash i}|u_i) \neq Q(\boldsymbol{u}^{\backslash i}|u_i)$, we still have that $Q^{\backslash j}(\boldsymbol{u}^{\backslash j}|u_j) = Q(\boldsymbol{u}^{\backslash j}|u_j)$ and

$$Q^{new}(\boldsymbol{u}^{\backslash i,j}|u_i, u_j) = Q(\boldsymbol{u}^{\backslash i,j}|u_i, u_j),$$

where $\boldsymbol{u}^{\backslash i,j}$ denotes all components of $\boldsymbol{u}$ except u_i, u_j. Let $Q^{\backslash j}(u_i) = N(h_i^{\backslash j}, \lambda_i^{\backslash j})$. Then,

$$\lambda_i^{\backslash j} = a_{i,i} + \frac{\pi_j a_{i,j}^2}{1 - \pi_j a_{j,j}}, \quad h_i^{\backslash j} = h_i + \frac{a_{i,j}\pi_j(h_j - m_j)}{1 - \pi_j a_{j,j}}.$$

It is clear that the ADF update remains the same as for the simple inclusion of i, however replacing $h_i^{\backslash i} = h_i$ by $h_i^{\backslash j}$ and $\lambda_i = a_{i,i}$ by $\lambda_i^{\backslash j}$. Note that the variance λ_i^{new} of $Q^{new}(u_i)$ is given by $\lambda_i^{new} = \lambda_i^{\backslash j}(1 - \nu_i \lambda_i^{\backslash j})$.

The representation discussed in Section C.3.1 can be updated as follows. Suppose, the position of j in I is l. Then, we have to replace $\boldsymbol{\Pi}_I^{1/2}$ by $\boldsymbol{\Pi}_I^{1/2} + (\pi_i^{1/2} - \pi_j^{1/2})\boldsymbol{\delta}_l\boldsymbol{\delta}_l^T$. Furthermore, $\boldsymbol{K}_I = \boldsymbol{I}_{I,\cdot}\boldsymbol{K}\boldsymbol{I}_{\cdot,I}$, but due to the change of I, $\boldsymbol{I}_{\cdot,I}$ has to be replaced by $\boldsymbol{I}_{\cdot,I} + (\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)\boldsymbol{\delta}_l^T$. Note that here, $\boldsymbol{\delta}_i, \boldsymbol{\delta}_j \in \mathbb{R}^n$, but $\boldsymbol{\delta}_l \in \mathbb{R}^d$.

[4] Our present implementation does not include such packets.

Therefore, $\boldsymbol{B}^{new} = \boldsymbol{I} + \boldsymbol{Q}^T\boldsymbol{K}\boldsymbol{Q}$ with

$$\boldsymbol{Q} = \left(\boldsymbol{I}_{\cdot,I} + (\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)\boldsymbol{\delta}_l^T\right)\left(\boldsymbol{\Pi}_I^{1/2} + (\pi_i^{1/2} - \pi_j^{1/2})\boldsymbol{\delta}_l\boldsymbol{\delta}_l^T\right).$$

Noting that $\boldsymbol{I}_{\cdot,I}\boldsymbol{\delta}_l = \boldsymbol{\delta}_j$ and $\boldsymbol{\Pi}_I^{1/2}\boldsymbol{\delta}_l = \pi_j^{1/2}\boldsymbol{\delta}_l$, we arrive at

$$\boldsymbol{Q} = \boldsymbol{I}_{\cdot,I}\boldsymbol{\Pi}_I^{1/2} + \left(\pi_i^{1/2}\boldsymbol{\delta}_i - \pi_j^{1/2}\boldsymbol{\delta}_j\right)\boldsymbol{\delta}_l^T.$$

It is now easy to see that if we set $\boldsymbol{v} = \boldsymbol{\Pi}_I^{1/2}\boldsymbol{I}_{I,\cdot}\boldsymbol{K}(\pi_i^{1/2}\boldsymbol{\delta}_i - \pi_j^{1/2}\boldsymbol{\delta}_j) = \boldsymbol{\Pi}_I^{1/2}(\pi_i^{1/2}\boldsymbol{K}_{I,i} - \pi_j^{1/2}\boldsymbol{K}_{I,j})$, then

$$\boldsymbol{B}^{new} - \boldsymbol{B} = \boldsymbol{\delta}_l\boldsymbol{v}^T + \boldsymbol{v}\boldsymbol{\delta}_l^T + \eta\boldsymbol{\delta}_l\boldsymbol{\delta}_l^T,$$

where $\eta = \pi_i\boldsymbol{K}_{i,i} + \pi_j\boldsymbol{K}_{j,j} - 2\pi_i^{1/2}\pi_j^{1/2}\boldsymbol{K}_{i,j} > 0$. Now, with $\tilde{\boldsymbol{\delta}}_l = \eta^{1/2}\boldsymbol{\delta}_l$, $\tilde{\boldsymbol{v}} = \eta^{-1/2}\boldsymbol{v}$ we arrive at

$$\boldsymbol{B}^{new} - \boldsymbol{B} = \left(\tilde{\boldsymbol{\delta}}_l + \tilde{\boldsymbol{v}}\right)\left(\tilde{\boldsymbol{\delta}}_l + \tilde{\boldsymbol{v}}\right)^T - \tilde{\boldsymbol{v}}\tilde{\boldsymbol{v}}^T.$$

We can now use the technique described in Section A.2.2 in order to update $\boldsymbol{L}$.

Denote $\boldsymbol{\Delta} = \mathrm{diag}(\pi_i, -\pi_j)$, the changes in $\boldsymbol{\Pi}$, and $\boldsymbol{\gamma} = (\pi_i m_i, -\pi_j m_j)^T$, the changes in $\boldsymbol{\Pi}\boldsymbol{m}$. Note that

$$\boldsymbol{A}^{new} = \boldsymbol{A} - \boldsymbol{A}\boldsymbol{I}_{\cdot,\{ij\}}\boldsymbol{\Delta}^{1/2}\boldsymbol{P}^{-1}\boldsymbol{\Delta}^{1/2}\boldsymbol{I}_{\{ij\},\cdot}\boldsymbol{A}, \quad \boldsymbol{P} = \boldsymbol{I} + \boldsymbol{\Delta}^{1/2}\boldsymbol{A}_{\{ij\}}\boldsymbol{\Delta}^{1/2}, \quad \text{(C.6)}$$

therefore $\boldsymbol{A}_{\{ij\}}^{new} = \boldsymbol{A}_{\{ij\}} - \boldsymbol{A}_{\{ij\}}\boldsymbol{\Delta}^{1/2}\boldsymbol{P}^{-1}\boldsymbol{\Delta}^{1/2}\boldsymbol{A}_{\{ij\}} = \boldsymbol{A}_{\{ij\}}\boldsymbol{\Delta}^{1/2}\boldsymbol{P}^{-1}\boldsymbol{\Delta}^{-1/2}$. Thus, by using the definition of $\boldsymbol{h}$ (see (4.6)), we arrive at

$$\boldsymbol{h}^{new} = \boldsymbol{h} + \boldsymbol{A}_{\cdot,\{ij\}}\boldsymbol{\eta}, \quad \boldsymbol{\eta} = \boldsymbol{\Delta}^{1/2}\boldsymbol{P}^{-1}\left(\boldsymbol{\Delta}^{-1/2}\boldsymbol{\gamma} - \boldsymbol{\Delta}^{1/2}\boldsymbol{h}_{\{ij\}}\right). \quad \text{(C.7)}$$

Finally, $\boldsymbol{M}$ is updated in two steps. First, compute

$$\boldsymbol{M}' = \boldsymbol{L}^{-1}\left(\boldsymbol{\Pi}_I^{1/2} + (\pi_i^{1/2} - \pi_j^{1/2})\boldsymbol{\delta}_l\boldsymbol{\delta}_l^T\right)\boldsymbol{K}_{I,\cdot} = \boldsymbol{M} + (\pi_i^{1/2} - \pi_j^{1/2})\left(\boldsymbol{L}^{-1}\boldsymbol{\delta}_l\right)\boldsymbol{K}_{l,\cdot}.$$

Second, note that the new Cholesky factor can be written as $\boldsymbol{L}\boldsymbol{L}^{(1)}\boldsymbol{L}^{(2)}$, but the factors $\boldsymbol{L}^{(k)}$ have a special structure, allowing for back-substitutions in $O(d)$. Therefore, the new $\boldsymbol{M}$ matrix can be computed from $\boldsymbol{M}'$ in $O(n\,d)$.

Note that in order to an exchange move, we require the element $a_{i,j}$ of $\boldsymbol{A}$ which can be computed from $\boldsymbol{M}$ if $\boldsymbol{K}_{i,j}$ is known. If enough memory is available, $\boldsymbol{K}_{I,\cdot}$ could be stored alongside $\boldsymbol{M}$. If the algorithm is at a stage for which further

new inclusions are not required, it is sensible to convert from $\boldsymbol{M}$ into an explicit representation of $\boldsymbol{A}$ which can then be updated using the Woodbury formula[5]. Finally, if kernel evaluations are inexpensive, the required elements of $\boldsymbol{K}$ can be re-evaluated whenever needed.

The differential entropy score can be computed using Lemma C.1 twice, namely since $Q^{new}(\boldsymbol{u}^{\backslash i}|u_i) = Q^{\backslash j}(\boldsymbol{u}^{\backslash i}|u_i)$ and $Q^{\backslash j}(\boldsymbol{u}^{\backslash j}|u_j) = Q(\boldsymbol{u}^{\backslash j}|u_j)$, we have

$$\begin{aligned}\Delta^{\mathrm{Ent}}_{(j,i)} &= \mathrm{H}[Q^{new}(u_i)] - \mathrm{H}[Q^{\backslash j}(u_i)] + \mathrm{H}[Q^{\backslash j}(u_j)] - \mathrm{H}[Q(u_j)] \\ &= \frac{1}{2}\log\left(\frac{\lambda_i^{new}\,(\boldsymbol{A}^{\backslash j})_{j,j}}{\lambda_i^{\backslash j}\,a_{j,j}}\right) = \frac{1}{2}\log\left(1 - \lambda_i^{\backslash j}\nu_i\right) - \frac{1}{2}\log(1 - a_{j,j}\pi_j).\end{aligned}$$

For the information gain score, we apply Lemma C.1 once more to obtain

$$-\mathrm{D}[Q^{new}(\boldsymbol{u}) \,\|\, Q(\boldsymbol{u})] = -\mathrm{D}[Q^{new}(u_i, u_j) \,\|\, Q(u_i, u_j)].$$

Here, $Q(u_i, u_j) = N(\boldsymbol{h}_{\{ij\}}, \boldsymbol{A}_{\{ij\}})$ and $Q^{new}(u_i, u_j) = N(\boldsymbol{h}^{new}_{\{ij\}}, \boldsymbol{A}^{new}_{\{ij\}})$, the latter are given by (C.6) and (C.7). Now, using (A.17), we arrive at

$$\Delta^{\mathrm{Info}}_{(j,i)} = -\frac{1}{2}\left(\log|\boldsymbol{P}| + \mathrm{tr}\,\boldsymbol{P}^{-1} + \boldsymbol{\eta}^T\boldsymbol{A}_{\{ij\}}\boldsymbol{\eta} - 2\right).$$

C.3.3 Model Selection Criterion and Gradient

In Section 4.5.2, we suggested a model selection criterion for IVM which works together efficiently with randomised greedy selection (RGS). Let $L \subset \{1, \ldots, n\}$ with $I \subset L$ such that a buffer of size d-by-$|L|$ can be stored. In order to compute the criterion and its gradient, we have to evaluate $\boldsymbol{M}_{\cdot,L}$ which is $O(|L|\,d^2)$ from scratch, but if we include patterns into L which have the largest stubs in the RGS cache, the evaluation can be significantly cheaper. A strategy for choosing L is discussed below.

The criterion is defined in the same way as $\mathcal{Q}^{MF}$ for PLV (see Section 4.5.1), but the true likelihood $P(\boldsymbol{y}|\boldsymbol{u})$ is replaced by $P(\boldsymbol{y}_L|\boldsymbol{u}_L)$. This leads to

$$\mathcal{G}^L = \mathrm{E}_Q\left[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)\right] + \mathrm{D}[Q(\boldsymbol{u}_I) \,\|\, P(\boldsymbol{u}_I)],$$

[5]This may be less numerically stable than the Cholesky updates here.

with

$$\mathrm{D}[Q(\boldsymbol{u}_I) \,\|\, P(\boldsymbol{u}_I)] = \frac{1}{2}\left(\log|\boldsymbol{B}| + \operatorname{tr}\boldsymbol{B}^{-1} + \|\boldsymbol{\beta}\|^2 - \left\|\boldsymbol{L}^{-T}\boldsymbol{\beta}\right\|^2 - d\right).$$

With $\boldsymbol{q} = \operatorname{diag}\boldsymbol{A} = \operatorname{diag}(\boldsymbol{K} - \boldsymbol{M}^T\boldsymbol{M})$ we have

$$\mathrm{E}_Q[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)] = -\sum_{i\in L}\mathrm{E}_Q[\log P(y_i|u_i)], \quad Q(u_i) = N(h_i, q_i).$$

Define $\boldsymbol{c}$ via $c_i = \mathrm{E}_Q[\log P(y_i|u_i)]$ for later use. We have $\boldsymbol{\beta} = \boldsymbol{L}^{-1}\boldsymbol{\Pi}_I^{-1/2}\boldsymbol{b}_I$ and $\boldsymbol{B} = \boldsymbol{I} + \boldsymbol{\Pi}_I^{1/2}\boldsymbol{K}_I\boldsymbol{\Pi}_I^{1/2}$, thus $d\boldsymbol{B} = \boldsymbol{\Pi}_I^{1/2}(d\boldsymbol{K}_I)\boldsymbol{\Pi}_I^{1/2}$ for a variation $d\boldsymbol{K}$. If we define

$$\boldsymbol{v} = \boldsymbol{L}^{-T}\boldsymbol{\beta}, \quad \boldsymbol{e}^{(1)} = \boldsymbol{\Pi}_I^{1/2}\boldsymbol{B}^{-1}\boldsymbol{v}, \qquad \boldsymbol{e}^{(2)} = \boldsymbol{\Pi}_I^{1/2}\boldsymbol{v},$$

then some algebra gives

$$d\log|\boldsymbol{B}| + d\operatorname{tr}\boldsymbol{B}^{-1} = \operatorname{tr}\boldsymbol{B}^{-1}\boldsymbol{\Pi}_I^{1/2}(d\boldsymbol{K}_I)\left(\boldsymbol{\Pi}_I^{1/2} - \boldsymbol{B}^{-1}\boldsymbol{\Pi}_I^{1/2}\right)^T$$

and

$$d\|\boldsymbol{\beta}\|^2 - d\left\|\boldsymbol{L}^{-T}\boldsymbol{\beta}\right\|^2 = -\boldsymbol{e}^{(2)T}(d\boldsymbol{K}_I)\boldsymbol{e}^{(2)} + \boldsymbol{e}^{(1)T}(d\boldsymbol{K}_I)\boldsymbol{e}^{(2)} + \boldsymbol{e}^{(2)T}(d\boldsymbol{K}_I)\boldsymbol{e}^{(1)}.$$

Furthermore, $\boldsymbol{h}_L = (\boldsymbol{M}_{\cdot,L})^T\boldsymbol{\beta} = \boldsymbol{K}_{L,I}\boldsymbol{\Pi}_I^{1/2}\boldsymbol{L}^{-T}\boldsymbol{\beta}$, so that

$$d\boldsymbol{h}_L = (d\boldsymbol{K}_{I,L})^T\boldsymbol{e}^{(2)} - \boldsymbol{E}^T(d\boldsymbol{K}_I)\boldsymbol{e}^{(2)}$$

with

$$\boldsymbol{E} = \boldsymbol{\Pi}_I^{1/2}\boldsymbol{L}^{-T}\boldsymbol{M}_{\cdot,L}.$$

Also, $\boldsymbol{q}_L = \operatorname{diag}(\boldsymbol{K}_L - \boldsymbol{M}_{\cdot,L}^T\boldsymbol{M}_{\cdot,L})$, so

$$d\boldsymbol{q}_L = d(\operatorname{diag}\boldsymbol{K})_L - 2\operatorname{diag}\boldsymbol{E}^T(d\boldsymbol{K}_{I,L}) + \operatorname{diag}\boldsymbol{E}^T(d\boldsymbol{K}_I)\boldsymbol{E}. \tag{C.8}$$

Next,

$$d\mathrm{E}_Q[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)] = -\sum_{i\in L}\mathrm{E}_Q\left[(\log P(y_i|u_i))d\log Q(u_i)\right].$$

With $a_i = \mathrm{E}_Q[(\log P(y_i|u_i))(u_i - h_i)/q_i]$ and $b_i = \mathrm{E}_Q[(\log P(y_i|u_i))((u_i - h_i)/q_i)^2]$, furthermore $d_i = (1/2)(c_i/q_i - b_i)$ we have

$$d\mathrm{E}_Q[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)] = \boldsymbol{d}_L^T(d\boldsymbol{q}_L) - \boldsymbol{a}_L^T d\boldsymbol{h}_L. \tag{C.9}$$

The computation of $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ can be done using numerical quadrature.

Since we only need an inner product with $d\boldsymbol{q}_L$, we can use the relation

$$\boldsymbol{u}^T \operatorname{diag} \boldsymbol{A}^T \boldsymbol{D} \boldsymbol{B} = \operatorname{tr} \boldsymbol{D}^T \left(\boldsymbol{A} (\operatorname{diag} \boldsymbol{u}) \boldsymbol{B}^T \right) \tag{C.10}$$

for matrices $\boldsymbol{A}, \boldsymbol{D}, \boldsymbol{B}$ and vector $\boldsymbol{u}$. Namely, with

$$\tilde{\boldsymbol{E}} = \boldsymbol{E} \left(\operatorname{diag} \boldsymbol{d}_L \right), \quad \tilde{\boldsymbol{F}} = \boldsymbol{E} \left(\operatorname{diag} \boldsymbol{d}_L \right) \boldsymbol{E}^T$$

we have

$$\begin{aligned} d\mathrm{E}_Q[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)] = \boldsymbol{d}_L^T d(\operatorname{diag} \boldsymbol{K})_L - \operatorname{tr} \left(2\tilde{\boldsymbol{E}} + \boldsymbol{e}^{(2)} \boldsymbol{a}_L^T \right)^T (d\boldsymbol{K}_{I,L}) \\ + \operatorname{tr} \tilde{\boldsymbol{F}} (d\boldsymbol{K}_I) + (\boldsymbol{E}\boldsymbol{a}_L)^T (d\boldsymbol{K}_I) \boldsymbol{e}^{(2)}. \end{aligned}$$

If the noise model depends on hyperparameters, a variation of these results in

$$d\mathrm{E}_Q[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)] = -\sum_{i \in L} \mathrm{E}_Q[d \log P(y_i|u_i)].$$

However, if the hyperparameter (α, say) is a location parameter in the sense that $(d/d\alpha) \log P(y_i|u_i) = (d/du_i) \log P(y_i|u_i)$, we can use partial integration to see that

$$\mathrm{E}_Q \left[\frac{d \log P(y_i|u_i)}{d\alpha} \right] = \mathrm{E}_Q \left[\frac{u_i - h_i}{q_i} \log P(y_i|u_i) \right] = a_i,$$

i.e. $(d/d\alpha) \mathrm{E}_Q[-\log P(\boldsymbol{y}_L|\boldsymbol{u}_L)] = -\boldsymbol{a}_L^T \boldsymbol{1}$.

The dominating precomputations are $\boldsymbol{M}_{\cdot,L}$, $\boldsymbol{E}$ and $\tilde{\boldsymbol{F}}$, the latter two require $O(|L|\, d^2)$ (the matrix $\tilde{\boldsymbol{E}}$ overwrites $\boldsymbol{M}_{\cdot,L}$). The former can be computed efficiently by completing stubs stored in the RGS cache (see Section C.3.1.1). To this end, we form L from the patterns with the largest stubs in this cache, given that L also contains I. As discussed in Section 4.5.2, the objectives for keeping a pattern's stub in the cache and having it in L are virtually identical. Given the precomputations, the computational cost is $O(n\, d)$ per gradient component. If the cost for the precomputation violates resource constraints, the dependence of $\boldsymbol{q}$ on $\boldsymbol{K}$ may be ignored to obtain an approximate gradient, although it is not clear whether criterion and approximate gradient can still be fed into a custom optimiser which typically requires that finite differences converge against gradients.

Our implementation requires one d-by-$|L|$ matrix buffer by allowing for the derivative matrices of $\boldsymbol{K}_{I,L}$ to be computed on the fly. It also manages to complete $\boldsymbol{M}_{\cdot,L}$ from the stubs available from RGS *without* having to allocate a second large buffer at any moment.

C.4 Derivations for Projected Latent Variables

In this section, we collect derivations required to develop the projected latent variables (PLV) scheme introduced in Section 4.4.2.

C.4.1 Site Approximation Updates. Point Inclusions

Recall that $\boldsymbol{\gamma} = \mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I] = \boldsymbol{V}^T\boldsymbol{L}^{-1}\boldsymbol{u}_I$. We have $\mathrm{E}_Q[\boldsymbol{\gamma}] = \boldsymbol{\mu}$ and $\mathrm{Var}_Q[\boldsymbol{\gamma}] = \boldsymbol{P}^T\mathrm{Var}_Q[\boldsymbol{u}_I]\boldsymbol{P} = \boldsymbol{V}^T\boldsymbol{M}^{-1}\boldsymbol{V}$, therefore $\mathrm{Var}_Q[\gamma_i] = q_i$. $Q(\boldsymbol{u})$ is given by (4.12). We mark variables *after* an update (or inclusion) by a prime (e.g., $Q \to Q'$).

Due to the nondiagonal likelihood approximation, the update of the site parameters for a point $i \notin I$ is slightly more involved than in the IVM case. If $Q^{\backslash i}(\boldsymbol{u}) \propto Q(\boldsymbol{u})N^U(\gamma_i| - b_i, -\pi_i)$ and $\hat{P}(\boldsymbol{u}) \propto t_i(u_i)Q^{\backslash i}(\boldsymbol{u})$, the site parameters have to be adjusted such that $Q'(\gamma_i)$ has the same mean and variance as $\hat{P}(\gamma_i)$. If

$$\tilde{t}_i(\gamma_i) = \int t_i(u_i)N(u_i|\gamma_i, l_i^2), \quad l_i^2 = \boldsymbol{K}_{i,i} - p_i,$$

then $\hat{P}(\gamma_i) \propto \tilde{t}_i(\gamma_i)Q^{\backslash i}(\gamma_i)$ since we have $Q^{\backslash i}(u_i|\boldsymbol{u}_I) = P(u_i|\boldsymbol{u}_i) = N(u_i|\gamma_i, l_i^2)$. $\hat{P}(\gamma_i)$ is a tilted Gaussian, we can use the derivation in Section C.1.2 if we substitute u_i for γ_i and $t_i(u_i)$ for $\tilde{t}_i(\gamma_i)$. To this end, we require $Q^{\backslash i}(\gamma_i)$ which is

$$Q^{\backslash i}(\gamma_i) = N\left(\gamma_i \,\middle|\, \mu_i + (q_i^{-1} - \pi_i)^{-1}(\pi_i\mu_i - b_i), (q_i^{-1} - \pi_i)^{-1}\right)$$

by an application of Lemma A.11. Note that we always have $q_i^{-1} - \pi_i > 0$. To see this, let $\boldsymbol{M}^{\backslash i} = \boldsymbol{I} + \boldsymbol{V}_{\cdot,\backslash i}\boldsymbol{\Pi}_{\backslash i}\boldsymbol{V}_{\cdot,\backslash i}^T$, which is positive definite. Then,

$$\pi_i^{-1} - q_i = \pi_i^{-1} - \boldsymbol{v}_i^T\left(\boldsymbol{M}^{\backslash i} + \pi_i\boldsymbol{v}_i\boldsymbol{v}_i^T\right)^{-1}\boldsymbol{v}_i = \left(\pi_i + (\pi_i\boldsymbol{v}_i)^T\boldsymbol{M}^{\backslash i}(\pi_i\boldsymbol{v}_i)\right)^{-1} > 0$$

if $\pi_i > 0$.

Note that the update now requires applying one-dimensional Gaussian expectations twice instead of only once in the diagonal case, which may raise problems if these expectations are done using numerical quadrature. For the probit classification noise model $t_i(u_i) = P(y_i|u_i) = \Phi(y_i(u_i + b))$, we arrive at the same formulae as in Section C.1.2, except that y_i has to be replaced by $y_i/\sqrt{1+l_i^2}$ (and $h_i^{\setminus i}$, λ_i by mean and variance of $Q^{\setminus i}(\gamma_i)$). It is easier to update the site parameters for $i \in I$, because $\gamma_i = u_i$ and we can use a normal EP update as described in Section 4.3: just replace $a_{i,i}$ by $l_i^2 + q_i$ and h_i by μ_i.

Since $Q(u_i) = N(\mu_i, q_i)$, we require μ_i, q_i. Recall that $\boldsymbol{C} = \boldsymbol{Q}^{-1}\boldsymbol{V}$ and note that $\mu_i = \boldsymbol{c}_i^T\boldsymbol{\beta}$, $q_i = \|\boldsymbol{c}_i\|^2$ where $\boldsymbol{c}_i = \boldsymbol{C}_{\cdot,i}$. We need to compute $\boldsymbol{c}_i$ which is $O(d^2)$. After the update, $\boldsymbol{Q}$ and related representation variables have to be updated. Suppose $\pi_i' = \pi_i + \Delta_\pi$, $b_i' = b_i + \Delta_b$. Then, $\boldsymbol{M}' = \boldsymbol{M} + \Delta_\pi \boldsymbol{v}_i\boldsymbol{v}_i^T$ with $\boldsymbol{v}_i = \boldsymbol{V}_{\cdot,i}$. Therefore, $\boldsymbol{Q}' = \boldsymbol{Q}\tilde{\boldsymbol{Q}}$ and back-substitutions with $\tilde{\boldsymbol{Q}}$ are $O(d)$ only (see Section A.2.2). $\tilde{\boldsymbol{Q}}$ can be computed in $O(d)$ given $\boldsymbol{c}_i$ which is available from the update. $\boldsymbol{Q}'$ itself is obtained from $\boldsymbol{Q}$ in $O(d^2)$. We can also update $\boldsymbol{\beta} = \boldsymbol{Q}^{-1}\boldsymbol{V}\boldsymbol{b}$ as

$$\boldsymbol{\beta}' = \tilde{\boldsymbol{Q}}^{-1}(\boldsymbol{\beta} + \Delta_b \boldsymbol{c}_i)$$

which is $O(d)$.

For reasons of numerical stability, $\boldsymbol{M}$ and $\boldsymbol{Q}$ should be "refreshed" now and then using the following procedure. After a fixed number $k < n$ of site parameter updates we have $\boldsymbol{M} = \boldsymbol{M}^{old} + \boldsymbol{V}(\Delta\boldsymbol{\Pi})\boldsymbol{V}^T$, where $\Delta\boldsymbol{\Pi} = \boldsymbol{\Pi} - \boldsymbol{\Pi}^{old}$. Here, at most k entries in $\Delta\boldsymbol{\Pi}$ are non-zero and $\Delta\boldsymbol{M} = \boldsymbol{V}(\Delta\boldsymbol{\Pi})\boldsymbol{V}^T$ can be computed in $O(k\,d^2)$. We then recompute $\boldsymbol{Q}$ in $O(d^3)$. In our implementation, $k = d$.

Let us finally see how to include a new point i into I. To this end, we have to update $\boldsymbol{L} \to \boldsymbol{L}' \in \mathbb{R}^{d+1,d+1}$, $\boldsymbol{V} \to \boldsymbol{V}' \in \mathbb{R}^{d+1,n}$, $\boldsymbol{p} \to \boldsymbol{p}'$ and $\boldsymbol{Q} \to \boldsymbol{Q}' \in \mathbb{R}^{d+1,d+1}$. First, $(\boldsymbol{L}')_{d+1,1\ldots d}^T = \boldsymbol{L}^{-1}\boldsymbol{K}_{I,i} = \boldsymbol{v}_i$, and $l_i = (\boldsymbol{L}')_{d+1,d+1} = (\boldsymbol{K}_{i,i} - \boldsymbol{v}_i^T\boldsymbol{v}_i)^{1/2} = (\boldsymbol{K}_{i,i} - p_i)^{1/2}$. Furthermore, $(\boldsymbol{V}')_{1\ldots d,\cdot} = \boldsymbol{V}$, and if $\boldsymbol{v} = (\boldsymbol{V}')_{d+1,\cdot}^T$, then $\boldsymbol{v} = l_i^{-1}(\boldsymbol{K}_{\cdot,i} - \boldsymbol{V}^T\boldsymbol{v}_i)$, and $\boldsymbol{p}' = \boldsymbol{p} + \boldsymbol{v} \circ \boldsymbol{v}$. Then, $\boldsymbol{z} = (\boldsymbol{Q}')_{d+1,1\ldots d} = \boldsymbol{Q}^{-1}\boldsymbol{V}\boldsymbol{\Pi}\boldsymbol{v}$ and $z = (\boldsymbol{Q}')_{d+1,d+1} = (1 + \boldsymbol{v}^T\boldsymbol{\Pi}\boldsymbol{v} - \boldsymbol{z}^T\boldsymbol{z})^{1/2}$. Finally, we update $\boldsymbol{\beta} \to \boldsymbol{\beta}'$ by adding the component $\beta_{d+1}' = z^{-1}(\boldsymbol{v}^T\boldsymbol{b} - \boldsymbol{\beta}^T\boldsymbol{z})$. The running time is dominated by the $O(n\,d)$ computations of $\boldsymbol{V}^T\boldsymbol{v}_i$ and $\boldsymbol{V}\boldsymbol{\Pi}\boldsymbol{v}$, furthermore we have to compute the kernel

matrix column $\boldsymbol{K}_{\cdot,i}$. The site parameters of i should be updated immediately after the inclusion using the procedure described above. The update of $\boldsymbol{Q}$ is unstable for very small Δ_π, and we recommend simply to skip such updates of $\boldsymbol{\Pi}$.

In the special case where the site parameters are fixed from the beginning (Gaussian likelihood, see end of Section 4.4.2), the vectors $\boldsymbol{\mu}$ and $\boldsymbol{q}$ (means and variances of Q marginals) can be maintained and updated explicitly like $\boldsymbol{p}$. The new row of $\boldsymbol{C}'$ is $\boldsymbol{c} = z^{-1}(\boldsymbol{v} - \boldsymbol{C}^T\boldsymbol{z})$, then $\boldsymbol{\mu}' = \boldsymbol{\mu} + \beta'_{d+1}\boldsymbol{c}$, $\boldsymbol{q}' = \boldsymbol{q} + \boldsymbol{c} \circ \boldsymbol{c}$. Since $\boldsymbol{C}^T\boldsymbol{z} = \boldsymbol{V}^T\boldsymbol{Q}^{-T}\boldsymbol{z}$, the matrix $\boldsymbol{C}$ need not be stored explicitly.

C.4.2 Information Gain Criterion

Suppose we want to score pattern i w.r.t. its value for inclusion into I. Recall from Section 4.4.2.1 that in order to obtain a cheap approximation for an expensive score $S(Q, Q^{new})$, where $Q^{new}(\boldsymbol{u})$ is the posterior approximation after inclusion of i, we can try to replace Q^{new} by $\tilde{Q}$ as defined in (4.15). Here, we show how the corresponding approximation to the negative information gain score can be computed:

$$\Delta_i = -\mathrm{D}[\tilde{Q}(\boldsymbol{u}) \,\|\, Q(\boldsymbol{u})].$$

As a first step, we have to update the site parameters to $\tilde{b}_i$, $\tilde{\pi}_i$ given that i is included into I. This is done by an ADF update, but it is a subtle point to note that the outcome is general different from the update described in Section C.4.1, because $\tilde{Q}$ has a likelihood approximation which depends on u_i. The cavity marginal $Q^{\setminus i}(u_i)$ is given by

$$Q^{\setminus i}(u_i) = N\left(u_i \,\middle|\, \mu_i + (q_i^{-1} - \pi_i)^{-1}(\pi_i\mu_i - b_i), (q_i^{-1} - \pi_i)^{-1} + l_i^2\right),$$

derived in the same way as in Section C.4.1 and noting that $Q^{\setminus i}(u_i|\boldsymbol{u}_I) = P(u_i|\boldsymbol{u}_I) = N(u_i|\gamma_i, l_i^2)$. The ADF update is then done as in Section C.1.2, rendering new values $\tilde{b}_i$, $\tilde{\pi}_i$. We denote the old values by b_i, π_i.

Now, $\tilde{Q}$ is obtained from Q by multiplying with $N^U(-b_i|\gamma_i, -\pi_i)$ and $N^U(\tilde{b}_i|u_i, \tilde{\pi}_i)$, then renormalising. Denote the normalisation constant by Z_i. In the sequel, we

will use $\langle\cdot\rangle$ to denote expectations over $\tilde{Q}$ and $\langle\cdot\,|\,\cdot\rangle$ for conditional expectations based on $\tilde{Q}$. We can first marginalise over $\boldsymbol{u}_{\setminus I'}$:

$$\Delta_i = \log Z_i + \frac{1}{2}\left(\tilde{\pi}_i\langle(\tilde{m}_i - u_i)^2\rangle - \pi_i\langle(m_i-\gamma_i)^2\rangle\right) \tag{C.11}$$

(recall that $m_i = b_i/\pi_i$). In order to compute Z_i, we first marginalise over u_i. Let $\tau_i = (1 + l_i^2\tilde{\pi}_i)^{-1}$ and $\Delta_b = \tau_i\tilde{b}_i - b_i,\ \Delta_\pi = \tau_i\tilde{\pi}_i - \pi_i,\ \Delta_m = \Delta_b/\Delta_\pi$ Then,

$$Z_i = \tau_i^{1/2}\exp\left(\frac{1}{2}\Delta_b\Delta_m - \frac{1}{2}\tilde{m}_i\tilde{b}_i(\tau_i - 1)\right)\mathrm{E}_{\gamma_i\sim Q}\left[N^{UC}\left(\gamma_i\,\middle|\,\Delta_m,\Delta_\pi\right)\right].$$

Recall that $Q(\gamma_i) = N(\mu_i, q_i)$, and define $\rho_i = (1 + \Delta_\pi q_i)^{-1}$. We can now use (A.16) and some algebra to obtain

$$\log Z_i = \frac{1}{2}\left(\log\tau_i + \Delta_m\Delta_b - \tilde{m}_i\tilde{b}_i(\tau_i - 1) + \log\rho_i - \Delta_\pi\rho_i(\mu_i - \Delta_m)^2\right). \tag{C.12}$$

Now to the second part of (C.11). We have $\langle(\tilde{m}_i - u_i)^2\rangle = \langle\langle(\tilde{m}_i - u_i)^2|\boldsymbol{u}_I\rangle\rangle = \langle(\tilde{m}_i - \langle u_i|\boldsymbol{u}_I\rangle)^2\rangle + \langle\mathrm{Var}(u_i|\boldsymbol{u}_I)\rangle$. Now, $\tilde{Q}(u_i|\boldsymbol{u}_I) \propto N^U(u_i|\tilde{b}_i,\tilde{\pi}_i)P(u_i|\boldsymbol{u}_I)$, i.e.

$$\langle u_i|\boldsymbol{u}_I\rangle = (\tilde{\pi}_i + l_i^{-2})^{-1}(\tilde{b}_i + l_i^{-2}\gamma_i), \quad \mathrm{Var}(u_i|\boldsymbol{u}_I) = (\tilde{\pi}_i + l_i^{-2})^{-1}.$$

Therefore, $\langle(\tilde{m}_i - u_i)^2\rangle = \tau_i^2\langle(\tilde{m}_i - \gamma_i)^2\rangle + l_i^2\tau_i$, and $\langle(m_i-\gamma_i)^2\rangle = (m_i - \langle\gamma_i\rangle)^2 + \mathrm{Var}(\gamma_i)$. From the application of (A.16) we know that $\mathrm{Var}(\gamma_i) = q_i\rho_i$ and $\langle\gamma_i\rangle = \Delta_m + \rho_i(\mu_i - \Delta_m)$, therefore $\langle(x-\gamma_i)^2\rangle = q_i\rho_i + (x - \Delta_m - \rho_i(\mu_i - \Delta_m))^2$. This is sufficient for computing (C.11).

Simplifications are possible if $\tilde{b}_i = b_i,\ \tilde{\pi}_i = \pi_i$. In this case, $\Delta_\pi = (\tau_i - 1)\pi_i,\ \Delta_b = (\tau_i - 1)b_i$ and $\Delta_m = m_i$, thus

$$\langle(m_i - u_i)^2\rangle - \langle(m_i-\gamma_i)^2\rangle = l_i^2\tau_i + (\tau_i^2 - 1)\left(q_i\rho_i + \rho_i^2(\mu_i - m_i)^2\right).$$

Define

$$\xi_i = (1-\tau_i)\rho_i = \frac{1}{1 + l_i^{-2}\pi_i^{-1} - \pi_i q_i}.$$

Then, some very tedious algebra reveals

$$\begin{aligned}\Delta_i &= -\log(l_i\pi_i^{1/2}) + \frac{1}{2}\left(\log\xi_i + \xi_i\pi_i(1-\kappa_i)(m_i-\mu_i)^2 - \kappa_i + 2\right),\\ \kappa_i &= \xi_i\left(1 + 2\pi_i^{-1}l_i^{-2}\right).\end{aligned} \tag{C.13}$$

C.4.3 Extended Information Gain Criterion

Suppose we want to score $i \notin I$ for inclusion. Recall from Section 4.4.2.1 that we can obtain successively more accurate approximations to the full information gain criterion by allowing for couplings between i and points from H with $H \cap I' = \emptyset$. Let $H' = H \cup \{i\}$ and $h = |H|$. Recall that $\boldsymbol{\gamma} = \mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_I] = \boldsymbol{P}^T\boldsymbol{u}_I$. Denote by $\boldsymbol{P}'$ the projection matrix after inclusion of i, and $\boldsymbol{\gamma}' = \mathrm{E}[\boldsymbol{u}|\boldsymbol{u}_{I'}] = (\boldsymbol{P}')^T\boldsymbol{u}_{I'}$. From the definition of $\boldsymbol{P}$, we see that $\boldsymbol{P}'_{d+1,H} = l_i^{-1}\boldsymbol{v}_H^T$, where $\boldsymbol{v} = (\boldsymbol{V}'_{d+1,\cdot})^T$ is the new row of $\boldsymbol{V}'$ when including i. Furthermore, if $\boldsymbol{\nu} = (\boldsymbol{P}'_{d+1,H}, 1)^T \in \mathbb{R}^{h+1}$, then $\boldsymbol{\gamma}'_{H'} = \boldsymbol{\gamma}_{H'} + (u_i - \gamma_i)\boldsymbol{\nu}$. We also use $\boldsymbol{m}_{H'} = \boldsymbol{\Pi}_{H'}^{-1}\boldsymbol{b}_{H'}$ and $\tilde{\boldsymbol{m}}_{H'} = \tilde{\boldsymbol{\Pi}}_{H'}^{-1}\tilde{\boldsymbol{b}}_{H'}$, with the usual convention that $m_j = 0$ if $\pi_j = b_i = 0$. In Section 4.4.2.1, we defined $\tilde{Q}$ as being obtained from Q by multiplying with $N^{UC}(\boldsymbol{\gamma}_{H'}|\boldsymbol{m}_{H'}, -\boldsymbol{\Pi}_{H'})$ and $N^{UC}(\boldsymbol{\gamma}'_{H'}|\tilde{\boldsymbol{m}}_{H'}, \tilde{\boldsymbol{\Pi}}_{H'})$. Note that here, we allow for $\boldsymbol{b}_{H'}, \boldsymbol{\Pi}_{H'}$ to be different from $\tilde{\boldsymbol{b}}_{H'}, \tilde{\boldsymbol{\Pi}}_{H'}$. Finally, recall that $Q(\boldsymbol{\gamma}_{H'}) = N(\boldsymbol{\mu}_{H'}, \boldsymbol{\Sigma}_{H'})$ with $\boldsymbol{\Sigma}_{H'} = (\boldsymbol{V}^T\boldsymbol{M}^{-1}\boldsymbol{V})_{H'} = (\boldsymbol{C}_{\cdot,H'})^T\boldsymbol{C}_{\cdot,H'}$. The computation of the covariance matrix is $O(|H'|d^2)$ if $\boldsymbol{C}_{\cdot,H'}$ is not available, $O(|H'|^2 d)$ otherwise. The criterion derived in Section C.4.2 is obtained as special case $H = \emptyset$.

We use the same approach as in Section C.4.2. First, we integrate out $\boldsymbol{u}_{\setminus I'}$, then determine the log partition function Z_i for $\tilde{Q}$ and the parameters for $\tilde{Q}(u_i|\boldsymbol{u}_I)$ and $\tilde{Q}(\boldsymbol{\gamma}_{H'})$. To compute Z_i, we first marginalise over u_i. Again, $Q(u_i|\boldsymbol{u}_I) = N^{UC}(u_i|\gamma_i, l_i^{-2})(2\pi l_i^2)^{-1/2}$. Second,

$$N^{UC}(\tilde{\boldsymbol{m}}_{H'}|\boldsymbol{\gamma}'_{H'}, \tilde{\boldsymbol{\Pi}}_{H'}) = N^{UC}(u_i\boldsymbol{\nu} \,|\, \tilde{\boldsymbol{m}}_{H'} - \boldsymbol{\gamma}_{H'} + \gamma_i\boldsymbol{\nu}, \tilde{\boldsymbol{\Pi}}_{H'}),$$

With

$$\begin{aligned}
&\boldsymbol{a} = \tilde{\boldsymbol{m}}_{H'} - \boldsymbol{\gamma}_{H'} + \gamma_i\boldsymbol{\nu}, \quad b = \boldsymbol{\nu}^T\tilde{\boldsymbol{\Pi}}_{H'}\boldsymbol{\nu}, \quad a = b^{-1}\boldsymbol{\nu}^T\tilde{\boldsymbol{\Pi}}_{H'}\boldsymbol{a}, \\
&c = \boldsymbol{a}^T\tilde{\boldsymbol{\Pi}}_{H'}\boldsymbol{a} - ba^2,
\end{aligned}$$

the factor becomes $N^{UC}(u_i|a, b)\exp(-c/2)$. Using (A.16), we obtain

$$Z_i = (1 + l_i^2 b)^{-1/2}\mathrm{E}_{\boldsymbol{u}_I \sim Q}\left[e^{-c/2}N^{UC}\left(a \,\middle|\, \gamma_i, (l_i^2 + b^{-1})^{-1}\right)N^{UC}\left(\boldsymbol{m}_{H'} \,\middle|\, \boldsymbol{\gamma}_{H'}, -\boldsymbol{\Pi}_{H'}\right)\right]. \tag{C.14}$$

The integrand is now a function of $\boldsymbol{\gamma}_{H'}$. To simplify c, define

$$\boldsymbol{e} = b^{-1}\tilde{\boldsymbol{\Pi}}_{H'}\boldsymbol{\nu}, \quad \varepsilon = \boldsymbol{e}^T\tilde{\boldsymbol{m}}_{H'} = b^{-1}\boldsymbol{\nu}^T\tilde{\boldsymbol{b}}_{H'}.$$

We have $c = \boldsymbol{a}^T(\tilde{\boldsymbol{\Pi}}_{H'} - b\boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{a}$ and $\boldsymbol{a} = \tilde{\boldsymbol{m}}_{H'} - (\boldsymbol{I} - \boldsymbol{\nu}\boldsymbol{\delta}_{h+1}^T)\boldsymbol{\gamma}_{H'}$. Note that $a = \boldsymbol{e}^T\boldsymbol{a}$ and $\boldsymbol{e}^T\boldsymbol{\nu} = 1$. Furthermore, $a - \gamma_i = \varepsilon - \boldsymbol{e}^T\boldsymbol{\gamma}_{H'}$. If $c = \boldsymbol{\gamma}_{H'}^T\boldsymbol{A}\boldsymbol{\gamma}_{H'} - 2\boldsymbol{f}^T\boldsymbol{\gamma}_{H'} + g$, i.e. $e^{-c/2} = N^U(\boldsymbol{\gamma}_{H'}|\boldsymbol{f}, \boldsymbol{A}, g)$, then by noting that $(\tilde{\boldsymbol{\Pi}}_{H'} - b\boldsymbol{e}\boldsymbol{e}^T)\boldsymbol{\nu} = \boldsymbol{0}$, i.e. $\tilde{\boldsymbol{\Pi}}_{H'} - b\boldsymbol{e}\boldsymbol{e}^T$ is not changed by multiplication with $\boldsymbol{I} - \boldsymbol{\nu}\boldsymbol{\delta}_{h+1}^T$, we obtain $\boldsymbol{A} = \tilde{\boldsymbol{\Pi}}_{H'} - b\boldsymbol{e}\boldsymbol{e}^T$, $\boldsymbol{f} = \tilde{\boldsymbol{\Pi}}_{H'}\tilde{\boldsymbol{m}}_{H'} - b\varepsilon\boldsymbol{e}$ and $g = \tilde{\boldsymbol{m}}_{H'}^T\tilde{\boldsymbol{\Pi}}_{H'}\tilde{\boldsymbol{m}}_{H'} - b\varepsilon^2$. For later use, we define

$$\begin{aligned}\boldsymbol{A}(x) &= \tilde{\boldsymbol{\Pi}}_{H'} - \boldsymbol{\Pi}_{H'} - x\boldsymbol{e}\boldsymbol{e}^T, \quad \boldsymbol{f}(x) = \tilde{\boldsymbol{b}}_{H'} - \boldsymbol{b}_{H'} - x\varepsilon\boldsymbol{e},\\ g(x) &= \tilde{\boldsymbol{m}}_{H'}^T\tilde{\boldsymbol{\Pi}}_{H'}\tilde{\boldsymbol{m}}_{H'} - \boldsymbol{m}_{H'}^T\boldsymbol{\Pi}_{H'}\boldsymbol{m}_{H'} - x\varepsilon^2.\end{aligned}$$

Note that

$$-2\log N^U(\boldsymbol{w}|\boldsymbol{f}(x), \boldsymbol{A}(x), g(x)) = -2\log N^U(\boldsymbol{w}|\boldsymbol{f}(0), \boldsymbol{A}(0), g(0)) - x(\varepsilon - \boldsymbol{e}^T\boldsymbol{w})^2. \tag{C.15}$$

The second factor in (C.14) is $N^{UC}(a|\gamma_i, \tau)$ with $\tau = (l_i^2 + b^{-1})^{-1}$, this is equal to $N^{UC}(\boldsymbol{e}^T\boldsymbol{\gamma}_{H'}|\varepsilon, \tau)$. Now, (C.15) implies that multiplying the first factor in (C.14) by the two remaining ones results in $\boldsymbol{A}$, $\boldsymbol{f}$, g being replaced by $\boldsymbol{A}(b-\tau)$, $\boldsymbol{f}(b-\tau)$, $g(b-\tau)$, i.e.

$$Z_i = (1 + l_i^2 b)^{-1/2}\mathrm{E}_{\boldsymbol{u}_I \sim Q}\left[N^U(\boldsymbol{\gamma}_{H'} \,|\, \boldsymbol{f}(b-\tau), \boldsymbol{A}(b-\tau), g(b-\tau))\right]. \tag{C.16}$$

We can now use (A.15) to obtain both Z_i and $\tilde{Q}(\boldsymbol{\gamma}_{H'})$. Let

$$\boldsymbol{R} = \boldsymbol{\Sigma}_{H'}^{-1} + \boldsymbol{A}(b-\tau), \quad \boldsymbol{r} = \boldsymbol{\Sigma}_{H'}^{-1}\boldsymbol{\mu}_{H'} + \boldsymbol{f}(b-\tau), \quad \tilde{\boldsymbol{\mu}} = \boldsymbol{R}^{-1}\boldsymbol{r}.$$

Then, $\tilde{Q}(\boldsymbol{\gamma}_{H'}) = N(\tilde{\boldsymbol{\mu}}, \boldsymbol{R}^{-1})$, and

$$\begin{aligned}\log Z_i = -\frac{1}{2}\big(&\log(1 + l_i^2 b) + \log|\boldsymbol{I} + \boldsymbol{\Sigma}_{H'}\boldsymbol{A}(b-\tau)| - \boldsymbol{r}^T\tilde{\boldsymbol{\mu}}\\ &+ \boldsymbol{\mu}_{H'}^T\boldsymbol{\Sigma}_{H'}^{-1}\boldsymbol{\mu}_{H'} + g(b-\tau)\big).\end{aligned}$$

Again, denote expectation over $\tilde{Q}$ by $\langle\cdot\rangle$. We have to compute (as second part of the criterion)

$$\left\langle -\log N^{UC}(\tilde{\boldsymbol{m}}_{H'}|\boldsymbol{\gamma}'_{H'}, \tilde{\boldsymbol{\Pi}}_{H'}) + \log N^{UC}(\boldsymbol{m}_{H'}|\boldsymbol{\gamma}_{H'}, \boldsymbol{\Pi}_{H'})\right\rangle. \tag{C.17}$$

As seen above, $\tilde{Q}(u_i|\boldsymbol{u}_I)$ has variance $(l_i^{-2} + b)^{-1}$ and mean $a + (1 + l_i^2 b)^{-1}(\gamma_i - a)$. We first condition on $\boldsymbol{u}_I$ and integrate out u_i. This affects $-\log N^{UC}(u_i|a, b)$ only

and results in

$$\left\langle \frac{b}{2}(1+l_i^2 b)^{-2}(\gamma_i - a)^2 + \frac{1}{2}(1+l_i^{-2}b^{-1})^{-1} \right\rangle = \left\langle \frac{1}{2}b^{-1}\tau^2(\varepsilon - \boldsymbol{e}^T\boldsymbol{\gamma}_{H'})^2 + \frac{1}{2}l_i^2\tau \right\rangle,$$

the first addend is just $-\log N^{UC}(\boldsymbol{e}^T\boldsymbol{\gamma}_{H'}|\varepsilon, b^{-1}\tau^2)$. Again, we can use (C.15) and merge this into the representation of c. Define $\sigma = 1-(1+l_i^2 b)^{-2}$, so that $b - b^{-1}\tau^2 = b\sigma$. Then, (C.17) evaluates to

$$\begin{aligned}&\left\langle -\log N^U(\boldsymbol{\gamma}_{H'}|\boldsymbol{f}(b\sigma), \boldsymbol{A}(b\sigma), g(b\sigma)) + \frac{1}{2}l_i^2\tau \right\rangle \\ =&\frac{1}{2}\left(\operatorname{tr}\boldsymbol{A}(b\sigma)\left(\boldsymbol{R}^{-1} + \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^T\right) - 2\boldsymbol{f}(b\sigma)^T\tilde{\boldsymbol{\mu}} + g(b\sigma) + l_i^2\tau\right).\end{aligned}$$

Finally, define $\rho = (1+b^{-1}l_i^{-2})^{-1} = l_i^2\tau$, and note that $b - \tau = b\rho$. Combining the two parts, we finally have

$$\begin{aligned}\Delta_i = \frac{1}{2}\Big(&-\log(1+l_i^2 b) - \log|\boldsymbol{I} + \boldsymbol{\Sigma}_{H'}\boldsymbol{A}(b\rho)| + \boldsymbol{r}^T\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu}_{H'}^T\boldsymbol{\Sigma}_{H'}^{-1}\boldsymbol{\mu}_{H'} \\ &+ \varepsilon^2 b(\rho-\sigma) + \operatorname{tr}\boldsymbol{A}(b\sigma)\left(\boldsymbol{R}^{-1} + \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^T\right) - 2\boldsymbol{f}(b\sigma)^T\tilde{\boldsymbol{\mu}} + \rho\Big).\end{aligned}$$

Using (C.15) we can bring this into a different form:

$$\begin{aligned}\Delta_i = \frac{1}{2}\Big(&-\log(1+l_i^2 b) - \log|\boldsymbol{\Sigma}_{H'}| - \log|\boldsymbol{R}| + \boldsymbol{r}^T\tilde{\boldsymbol{\mu}} \\ &- \boldsymbol{\mu}_{H'}^T\boldsymbol{\Sigma}_{H'}^{-1}\boldsymbol{\mu}_{H'} - g(b\rho) + \operatorname{tr}\boldsymbol{A}(b\sigma)\boldsymbol{R}^{-1} + \rho \\ &- 2\log N^U\left(\tilde{\boldsymbol{\mu}} \,\middle|\, \boldsymbol{f}(0), \boldsymbol{A}(0), g(0)\right) - b\sigma\left(\varepsilon - \boldsymbol{e}^T\tilde{\boldsymbol{\mu}}\right)^2\Big).\end{aligned} \tag{C.18}$$

C.4.3.1 Site Parameters Do Not Change

Note that if $\tilde{\boldsymbol{b}}_{H'} = \boldsymbol{b}_{H'}$, $\tilde{\boldsymbol{\Pi}}_{H'} = \boldsymbol{\Pi}_{H'}$, the criterion expression can be simplified considerably. Let $\alpha_1 = \boldsymbol{e}^T\boldsymbol{\Sigma}_{H'}\boldsymbol{e}$, $\alpha_2 = \boldsymbol{e}^T\boldsymbol{\mu}_{H'}$, furthermore

$$\xi_i = \frac{1}{\rho^{-1} - b\alpha_1},\ \kappa_i = \xi_i\left(1 + 2(bl_i^2)^{-1}\right).$$

Then, in the case $h \le d$, some tedious algebra gives

$$\Delta_i = -\log(l_i b^{1/2}) + \frac{1}{2}\left(\log\xi_i + b\xi_i(1-\kappa_i)(\varepsilon-\alpha_2)^2 - \kappa_i + 2\right). \tag{C.19}$$

From this expression, it is easy to recognise (C.13) as the special case $H = \emptyset$. Note that (C.19) is all we need in the case of GP regression with Gaussian noise, since the site parameters are fixed throughout the algorithm. It also serves as starting point for the general case, in which we start with $\tilde{\boldsymbol{b}}_{H'} = \boldsymbol{b}_{H'}$, $\tilde{\boldsymbol{\Pi}}_{H'} = \boldsymbol{\Pi}_{H'}$, then update $\tilde{\boldsymbol{b}}_{H'}$, $\tilde{\boldsymbol{\Pi}}_{H'}$ sequentially, as described below. Note that in order to compute (C.19), we do not need to compute $\boldsymbol{\Sigma}_{H'}$ or $\boldsymbol{\mu}_{H'}$ explicitly. It is sufficient to compute $\boldsymbol{u} = \boldsymbol{Q}^{-1}\boldsymbol{V}_{\cdot,H'}\boldsymbol{e} = \boldsymbol{C}_{\cdot,H'}\boldsymbol{e}$, since $\alpha_1 = \boldsymbol{u}^T\boldsymbol{u}$, $\alpha_2 = \boldsymbol{u}^T\boldsymbol{\beta}$. If $\boldsymbol{C}_{\cdot,H'}$ is not available, this costs $O(d^2)$ for the back-substitution which dominates the evaluation of (C.19) if $h < d$. We can identify (C.13) as special case in which $\boldsymbol{e} = 1$ and $\boldsymbol{u} = \boldsymbol{C}_{\cdot,i}$, thus $\alpha_1 = q_i$, $\alpha_2 = \mu_i$.

C.4.3.2 Updating Site Parameters

Before the criterion is evaluated, it is sensible to update the site parameters for H', using EP steps. Here, we describe a sequential scheme which updates Δ_i and associated variables (such as $\boldsymbol{e}$) alongside, which is necessary in order to compute the marginals required for the EP updates. Namely, for $j \in H'$ located at position l in H' we require the cavity marginal $\tilde{Q}^{\backslash j}(u_j)$, obtained by setting the corresponding elements in $\tilde{\boldsymbol{b}}$, $\tilde{\boldsymbol{\Pi}}$ to zero. For simplicity, we refer to this cavity marginal as $\tilde{Q}$. Recall that $\boldsymbol{\gamma}'_{H'} = \boldsymbol{\gamma}_{H'} + (u_i - \gamma_i)\boldsymbol{\nu}$, a Gaussian variable under $\tilde{Q}$. Recall that the mean of $\tilde{Q}(u_i|\boldsymbol{u}_I)$ is

$$b^{-1}\rho(l_i^{-2}\gamma_i + ba) = \gamma_i + \rho(a - \gamma_i),$$

the variance is $b^{-1}\rho$. Therefore, $\langle\boldsymbol{\gamma}'_{H'}\rangle = \tilde{\boldsymbol{\mu}} + \langle a - \gamma_i\rangle\rho\boldsymbol{\nu} = \tilde{\boldsymbol{\mu}} + \rho(\varepsilon - \boldsymbol{e}^T\tilde{\boldsymbol{\mu}})\boldsymbol{\nu}$. For $\mathrm{Var}[\boldsymbol{\gamma}'_{H'}]$, we use (A.2) and $\langle\boldsymbol{\gamma}'_{H'} \,|\, \boldsymbol{u}_I\rangle = (\boldsymbol{I} - \rho\boldsymbol{\nu}\boldsymbol{e}^T)\boldsymbol{\gamma}_{H'} + \varepsilon\rho\boldsymbol{\nu}$ to obtain

$$\mathrm{Var}[\boldsymbol{\gamma}'_{H'}] = \left(\boldsymbol{I} - \rho\boldsymbol{\nu}\boldsymbol{e}^T\right)\boldsymbol{R}^{-1}\left(\boldsymbol{I} - \rho\boldsymbol{e}\boldsymbol{\nu}^T\right) + b^{-1}\rho\boldsymbol{\nu}\boldsymbol{\nu}^T.$$

Recall from Section C.4.1 that for an ADF update we need the cavity marginal $\tilde{Q}(\gamma'_j)$. Since $\gamma'_j = \boldsymbol{\delta}_l^T\boldsymbol{\gamma}'_{H'}$, we can read off the marginal

$$\tilde{Q}(\gamma'_j) = N\left(\rho\varepsilon\nu_l + (\boldsymbol{\delta}_l - \rho\nu_l\boldsymbol{e})^T\tilde{\boldsymbol{\mu}},\, b^{-1}\rho\nu_l^2 + (\boldsymbol{\delta}_l - \rho\nu_l\boldsymbol{e})^T\boldsymbol{R}^{-1}(\boldsymbol{\delta}_l - \rho\boldsymbol{e})\right). \quad \text{(C.20)}$$

Recall that if $j \neq i$, we also require $l'^2_j = \boldsymbol{K}_{j,j} - p'_j$, where $p'_j = p_j + (l_i\nu_l)^2$ (recall that $\boldsymbol{\nu}_{1\ldots d} = l_i^{-1}\boldsymbol{v}_H$).

A sequential updating scheme could be implemented using the Woodbury formula (Lemma A.2), but for reasons of numerical stability, we use incremental Cholesky techniques instead (see Section A.2.2). Let

$$\boldsymbol{R}_0 = \boldsymbol{\Sigma}_{H'}^{-1} + \boldsymbol{A}(0) = \boldsymbol{L}_0 \boldsymbol{L}_0^T.$$

Since $\boldsymbol{R} = \boldsymbol{R}_0 - b\rho \boldsymbol{e}\boldsymbol{e}^T$, the Cholesky factor of $\boldsymbol{R}$ can be written as $\boldsymbol{L}_0\boldsymbol{L}_1$, where $\boldsymbol{L}_1$ is of a type discussed in Section A.2.2: back-substitutions with $\boldsymbol{L}_1$ are $O(h)$. Given $\boldsymbol{L}_0^{-1}\boldsymbol{e}$, we can compute $\boldsymbol{L}_1$ in $O(h)$. Factors of this type will be called *cheap*, they can be stored in $O(h)$. To ease the notation, we define

$$\boldsymbol{x}[\boldsymbol{A}] = \boldsymbol{A}^{-1}\boldsymbol{x}$$

for a vector $\boldsymbol{x}$ and a Cholesky factor $\boldsymbol{A}$. Note that $\boldsymbol{x}[\boldsymbol{A}\boldsymbol{B}] = \boldsymbol{x}[\boldsymbol{A}][\boldsymbol{B}]$, and if $\boldsymbol{A}$ is cheap, then $\boldsymbol{x}[\boldsymbol{A}]$ can be computed in linear time. Furthermore, let $\boldsymbol{r}_0 = \boldsymbol{r} + b\rho\varepsilon\boldsymbol{e} = \boldsymbol{\Sigma}_{H'}^{-1}\boldsymbol{\mu}_{H'} + \tilde{\boldsymbol{b}}_{H'} - \boldsymbol{b}_{H'}$. As a representation, we maintain the following variables up-to-date: $\boldsymbol{\delta}_m[\boldsymbol{L}_0]$, $m = 1, \ldots, h+1$, $\boldsymbol{e}[\boldsymbol{L}_0]$, $\boldsymbol{r}_0[\boldsymbol{L}_0]$, $\boldsymbol{f}(0)[\boldsymbol{L}_0]$, furthermore $\log|\boldsymbol{R}_0|$, $g(0)$.

The update procedure has two stages, both modify

$$\tilde{\boldsymbol{b}}_{H'} \to \tilde{\boldsymbol{b}}_{H'} + \Delta_b \boldsymbol{\delta}_l, \quad \tilde{\boldsymbol{\Pi}}_{H'} \to \tilde{\boldsymbol{\Pi}}_{H'} + \Delta_\pi \boldsymbol{\delta}_l \boldsymbol{\delta}_l^T.$$

Furthermore, define Δ_g such that $\tilde{\boldsymbol{b}}_{H'}^T \tilde{\boldsymbol{m}}_{H'} \to \tilde{\boldsymbol{b}}_{H'}^T \tilde{\boldsymbol{m}}_{H'} + \Delta_g$, i.e. $g(0) \to g(0) + \Delta_g$. This will change all associated variables, and we mark the corresponding new versions with a prime (for example, $\boldsymbol{e} \to \boldsymbol{e}'$). The Cholesky factor for $\boldsymbol{R}'$ is $\boldsymbol{L}_0'\boldsymbol{L}_1'$. Since

$$\boldsymbol{R}_0' = \boldsymbol{R}_0 + \Delta_\pi \boldsymbol{\delta}_l \boldsymbol{\delta}_l^T = \boldsymbol{L}_0 \left(\boldsymbol{I} + \Delta_\pi \boldsymbol{\delta}_l[\boldsymbol{L}_0]\boldsymbol{\delta}_l[\boldsymbol{L}_0]^T\right) \boldsymbol{L}_0^T,$$

we can compute $\boldsymbol{L}_0' = \boldsymbol{L}_0\tilde{\boldsymbol{L}}_0$, where $\tilde{\boldsymbol{L}}_0$ is cheap. This allows us to update the $\boldsymbol{x}[\boldsymbol{L}_0]$ variables. Since $\boldsymbol{R}' = \boldsymbol{R}_0' - b'\rho'\boldsymbol{e}'\boldsymbol{e}'^T$, we can compute the new (cheap) $\boldsymbol{L}_1'$ in $O(h)$, given $\boldsymbol{e}'[\boldsymbol{L}_0']$. We also define

$$\begin{aligned}
\boldsymbol{g}(\alpha, \beta) &= -\alpha\boldsymbol{e}' + \beta\boldsymbol{\delta}_l, \\
\boldsymbol{h}(\alpha, \beta) &= \boldsymbol{g}(\alpha, \beta)[\boldsymbol{L}_0'\boldsymbol{L}_1'] = -\alpha\boldsymbol{e}'[\boldsymbol{L}_0'\boldsymbol{L}_1'] + \beta\boldsymbol{\delta}_l[\boldsymbol{L}_0'\boldsymbol{L}_1'].
\end{aligned}$$

Note that $\boldsymbol{h}(\alpha_1,\beta_1)^T\boldsymbol{h}(\alpha_2,\beta_2) = \boldsymbol{g}(\alpha_1,\beta_1)^T(\boldsymbol{R}')^{-1}\boldsymbol{g}(\alpha_2,\beta_2)$. Also note that $\boldsymbol{h}(\alpha,\beta)$ can be computed in $O(h)$. Since $\boldsymbol{h}(\alpha,\beta)$ is required fairly often, it makes sense to store $-\boldsymbol{e}'[\boldsymbol{L}_0'\boldsymbol{L}_1'] = \boldsymbol{h}(1,0)$ and $\boldsymbol{\delta}_l[\boldsymbol{L}_0'\boldsymbol{L}_1'] = \boldsymbol{h}(0,1)$ explicitly.

It is easy to see that

$$b' = b + \Delta_\pi \nu_l^2, \quad \varepsilon' = (b')^{-1}(b\varepsilon + \Delta_b\nu_l),$$

furthermore

$$\boldsymbol{e}' = \alpha_e\boldsymbol{e} + \beta_e\boldsymbol{\delta}_l, \quad \alpha_e = b/b',\ \beta_e = \Delta_\pi\nu_l/b'.$$

Thus, $\boldsymbol{e}'[\boldsymbol{L}_0'] = \alpha_e\boldsymbol{e}[\boldsymbol{L}_0][\tilde{\boldsymbol{L}}_0] + \beta_e\boldsymbol{\delta}_l[\boldsymbol{L}_0']$, from which $\boldsymbol{L}_1'$ can be computed as mentioned above.

In the first stage, we compute the cavity marginal (C.20) in order to perform an EP update, i.e. $\Delta_b = -\tilde{b}_j,\ \Delta_\pi = -\tilde{\pi}_j$. If $\boldsymbol{w} = \boldsymbol{g}(\rho'\nu_l, 1)$, then we need $\boldsymbol{w}^T\tilde{\boldsymbol{\mu}}'$ and $\boldsymbol{w}^T(\boldsymbol{R}')^{-1}\boldsymbol{w}$. We have $\boldsymbol{w}[\boldsymbol{L}_0'\boldsymbol{L}_1'] = \boldsymbol{h}(\rho'\nu_l, 1)$. The inner product with $\tilde{\boldsymbol{\mu}}'$ is computed as described in the next paragraph. Given the cavity marginal, we do an EP step to obtain new values $\tilde{\pi}_j',\ \tilde{b}_j'$. Apart from these, we drop all values computed in the first stage. Only if $\tilde{\pi}_j'$ is significantly different from $\tilde{\pi}_j$, do we enter the second stage, where the representation is updated explicitly.

Inner products with $\tilde{\boldsymbol{\mu}}'$ are done as follows. Note that $\tilde{\boldsymbol{\mu}}' = (\boldsymbol{R}')^{-1}\boldsymbol{r}'$ and recall that $\boldsymbol{r}_0 = \boldsymbol{r} + b\rho\varepsilon\boldsymbol{e}$. Then,

$$\boldsymbol{r}'[\boldsymbol{L}_0'\boldsymbol{L}_1'] = \boldsymbol{r}_0'[\boldsymbol{L}_0'][\boldsymbol{L}_1'] + \boldsymbol{h}(b'\rho'\varepsilon', 0),$$

which is used to do inner products with $\tilde{\boldsymbol{\mu}}'$: $\boldsymbol{x}^T\tilde{\boldsymbol{\mu}}' = \boldsymbol{x}[\boldsymbol{L}_0'\boldsymbol{L}_1']^T\boldsymbol{r}'[\boldsymbol{L}_0'\boldsymbol{L}_1']$. $\boldsymbol{r}_0[\boldsymbol{L}_0]$ is updated as

$$\boldsymbol{r}_0'[\boldsymbol{L}_0'] = (\boldsymbol{r}_0 + \Delta_b\boldsymbol{\delta}_l)[\boldsymbol{L}_0'] = \boldsymbol{r}_0[\boldsymbol{L}_0][\tilde{\boldsymbol{L}}_0] + \Delta_b\boldsymbol{\delta}_l[\boldsymbol{L}_0'].$$

In the second stage, $\Delta_\pi = \tilde{\pi}_j' - \tilde{\pi}_j,\ \Delta_b = \tilde{b}_j' - \tilde{b}_j$ and Δ_g accordingly. We first compute $\tilde{\boldsymbol{L}}_0$ and $\boldsymbol{e}'[\boldsymbol{L}_0']$, $\boldsymbol{\delta}_m[\boldsymbol{L}_0']$, $m = 1,\ldots,h+1$ by back-substitutions, furthermore $\boldsymbol{L}_1'$ (see above). The criterion can be computed given the representation variables, so it suffices to update the latter. Convergence w.r.t. the criterion could be used as stopping criterion, but the update of the representation alone is somewhat cheaper than the full criterion computation. We have

$\boldsymbol{r}'_{0'} = \boldsymbol{r}_0 + \Delta_b \boldsymbol{\delta}_l$, $\boldsymbol{f}'(0) = \boldsymbol{f}(0) + \Delta_b \boldsymbol{\delta}_l$, so that in both cases the update works as

$$\boldsymbol{x}'[\boldsymbol{L}'_0] = \boldsymbol{x}[\boldsymbol{L}_0][\tilde{\boldsymbol{L}}_0] + \Delta_b \boldsymbol{\delta}_l[\boldsymbol{L}'_0].$$

Next, $\log|\boldsymbol{R}'_0| = \log|\boldsymbol{R}_0| + 2\log|\tilde{\boldsymbol{L}}_0|$ and $g'(0) = g(0) + \Delta_g$. We complete the second stage by updating $\tilde{b}_j$, $\tilde{\pi}_j$.

Finally, here is how to compute Δ'_i given the representation (using primed notation). First,

$$\log|\boldsymbol{R}'| = \log|\boldsymbol{R}'_0| + \log\left(1 - b'\rho' \left\|\boldsymbol{e}'[\boldsymbol{L}'_0]\right\|^2\right).$$

Next, $\boldsymbol{r}'^T \tilde{\boldsymbol{\mu}}' = \|\boldsymbol{r}'[\boldsymbol{L}'_0 \boldsymbol{L}'_1]\|^2$, as has been shown above. $g(b'\rho') = g'(0) - b'\rho'(\varepsilon')^2$, and

$$\operatorname{tr} \boldsymbol{A}'(b'\sigma')(\boldsymbol{R}')^{-1} = \operatorname{tr} \boldsymbol{A}'(0) \left(\operatorname{diag}(\boldsymbol{R}')^{-1}\right) - b'\sigma' \left\|\boldsymbol{h}(1,0)\right\|^2,$$

since $\boldsymbol{A}'(0)$ is diagonal. The elements of $\operatorname{diag}(\boldsymbol{R}')^{-1}$ are the squared norms of $\boldsymbol{\delta}_m[\boldsymbol{L}'_0 \boldsymbol{L}'_1]$, the latter vectors have to be computed for $m = 1, \ldots, h+1$. Next,

$$\tilde{\boldsymbol{\mu}}'^T \boldsymbol{A}'(0) \tilde{\boldsymbol{\mu}}' = \sum_{m=1}^{h+1} (\boldsymbol{A}'(0))_{m,m} \left(\tilde{\boldsymbol{\mu}}'^T \boldsymbol{\delta}_m\right)^2,$$

for which we require the $\boldsymbol{\delta}_m[\boldsymbol{L}'_0 \boldsymbol{L}'_1]$ once more. Finally, $\boldsymbol{f}'(0)^T \tilde{\boldsymbol{\mu}}'$ and $\boldsymbol{e}'^T \tilde{\boldsymbol{\mu}}'$ are computed as usual.

If $h < d$, the cost of updating the site parameters and evaluating Δ_i is dominated by the computation of $\boldsymbol{\Sigma}_{H'}$, which requires $h+1$ back-substitutions with $\boldsymbol{Q}$ (to compute $\boldsymbol{C}_{\cdot,H'}$). Depending on the update strategy of the concrete implementation, a larger part or the complete $\boldsymbol{C}_{\cdot,H'}$ may be available.

C.4.3.3 The Case $|H'| > d$

So far, we have silently assumed that $|H'| \leq d$. If this is not the case, the above derivation has to be modified as follows. We have $\boldsymbol{\gamma}_{H'} = \boldsymbol{P}^T_{\cdot,H'} \boldsymbol{u}_I$, with $Q(\boldsymbol{u}_I) = N(\boldsymbol{\mu}_I, \boldsymbol{\Sigma}_I)$, $\boldsymbol{\Sigma}_I = \boldsymbol{L}\boldsymbol{M}^{-1}\boldsymbol{L}^T$. Go as far as (C.16), but take the integrand as a function of $\boldsymbol{u}_I$:

$$Z_i = (1 + l_i^2 b)^{-1/2} \mathrm{E}_{\boldsymbol{u}_I \sim Q} \left[N^U(\boldsymbol{u}_I \,|\, \tilde{\boldsymbol{f}}(b-\tau), \tilde{\boldsymbol{A}}(b-\tau), g(b-\tau))\right],$$

with

$$\tilde{\boldsymbol{A}}(x) = \boldsymbol{P}_{\cdot,H'}\left(\tilde{\boldsymbol{\Pi}}_{H'} - \boldsymbol{\Pi}_{H'}\right)\boldsymbol{P}_{\cdot,H'}^T - x\tilde{\boldsymbol{e}}\tilde{\boldsymbol{e}}^T,$$
$$\tilde{\boldsymbol{f}}(x) = \boldsymbol{P}_{\cdot,H'}\left(\tilde{\boldsymbol{b}}_{H'} - \boldsymbol{b}_{H'}\right) - x\varepsilon\tilde{\boldsymbol{e}}, \quad \tilde{\boldsymbol{e}} = b^{-1}\boldsymbol{P}_{\cdot,H'}\tilde{\boldsymbol{\Pi}}_{H'}\boldsymbol{\nu}.$$

We now use $\tilde{Q}(\boldsymbol{u}_I) = N(\tilde{\boldsymbol{\mu}}, \boldsymbol{R}^{-1})$ instead of $\tilde{Q}(\boldsymbol{\gamma}_{I'})$ (the latter is degenerate), leading to

$$\boldsymbol{R} = \boldsymbol{\Sigma}_I^{-1} + \tilde{\boldsymbol{A}}(b\rho), \quad \boldsymbol{r} = \boldsymbol{\Sigma}_I^{-1}\boldsymbol{\mu}_I + \tilde{\boldsymbol{f}}(b\rho), \quad \tilde{\boldsymbol{\mu}} = \boldsymbol{R}^{-1}\boldsymbol{r}.$$

With these re-definitions, we have

$$\Delta_i = \frac{1}{2}\Big(-\log(1+l_i^2 b) - \log\left|\boldsymbol{I} + \boldsymbol{\Sigma}_I\tilde{\boldsymbol{A}}(b\rho)\right| + \boldsymbol{r}^T\boldsymbol{R}^{-1}\boldsymbol{r} - \boldsymbol{\mu}_I^T\boldsymbol{\Sigma}_I^{-1}\boldsymbol{\mu}_I$$
$$+ \varepsilon^2 b(\rho-\sigma) + \operatorname{tr}\tilde{\boldsymbol{A}}(b\sigma)\left(\boldsymbol{R}^{-1} + \tilde{\boldsymbol{\mu}}\tilde{\boldsymbol{\mu}}^T\right) - 2\tilde{\boldsymbol{f}}(b\sigma)^T\tilde{\boldsymbol{\mu}} + \rho\Big),$$

or equivalently

$$\Delta_i = \frac{1}{2}\Big(-\log(1+l_i^2 b) - \log|\boldsymbol{\Sigma}_I| - \log|\boldsymbol{R}| + \boldsymbol{r}^T\tilde{\boldsymbol{\mu}}$$
$$- \boldsymbol{\mu}_I^T\boldsymbol{\Sigma}_I^{-1}\boldsymbol{\mu}_I - g(b\rho) + \operatorname{tr}\tilde{\boldsymbol{A}}(b\sigma)\boldsymbol{R}^{-1} + \rho$$
$$- 2\log N^U\left(\tilde{\boldsymbol{\mu}} \,\middle|\, \tilde{\boldsymbol{f}}(0), \tilde{\boldsymbol{A}}(0), g(0)\right) - b\sigma\left(\varepsilon - \tilde{\boldsymbol{e}}^T\tilde{\boldsymbol{\mu}}\right)^2\Big).$$

The simpler form (C.19) is valid in this case if we re-define $\alpha_1 = \tilde{\boldsymbol{e}}^T\boldsymbol{\Sigma}_I\tilde{\boldsymbol{e}}$, $\alpha_2 = \tilde{\boldsymbol{e}}^T\boldsymbol{\mu}_I$. In fact, it is easy to see that the computation of the form (C.19) is exactly the same in both cases $h < d$ and $h \geq d$. Since $\tilde{\boldsymbol{e}} = b^{-1}\boldsymbol{L}^{-T}\boldsymbol{V}_{\cdot,H'}\tilde{\boldsymbol{\Pi}}_{H'}\boldsymbol{\nu}$, we can compute $\alpha_1 = \boldsymbol{u}^T\boldsymbol{u}$, $\alpha_2 = \boldsymbol{u}^T\boldsymbol{\beta}$ with

$$\boldsymbol{u} = \boldsymbol{Q}^{-1}\boldsymbol{L}^T\tilde{\boldsymbol{e}} = \boldsymbol{C}_{\cdot,H'}\boldsymbol{e}$$

at a cost of $O(h\,d)$ if $\boldsymbol{C}_{\cdot,H'}$ is given. If we denote the columns of $\boldsymbol{P}_{\cdot,H'}$ by $\boldsymbol{w}_m \in \mathbb{R}^d$, we can use the sequential updating scheme described above as it stands if we do the substitutions implied by the re-definitions and also replace $\boldsymbol{\delta}_m$ by $\boldsymbol{w}_m$ everywhere. The computation of $\boldsymbol{P}_{\cdot,H'}$ is $O(h\,d^2)$. Instead of keeping all $\boldsymbol{w}_m[\boldsymbol{L}_0]$ up-to-date, we maintain $\boldsymbol{L}_0$ explicitly which allows us to compute $\boldsymbol{w}_l[\boldsymbol{L}_0]$

directly in each iteration. Although the final criterion evaluation can be done more efficiently, we simply copy the scheme above, evaluating for example

$$\operatorname{tr} \tilde{\boldsymbol{A}}'(0)(\boldsymbol{R}')^{-1} = \sum_{m=1}^{h+1} (\tilde{\boldsymbol{\Pi}}_{H'} - \boldsymbol{\Pi}_{H'})_{m,m} \left\| \boldsymbol{w}'_m [\boldsymbol{L}'_0 \boldsymbol{L}'_1] \right\|^2$$

at a cost of $O(h\, d^2)$. A more efficient evaluation in $O(d^2)$ is possible if the representation is extended, but this is not considered here.

C.4.4 Gradient of Model Selection Criterion

Recall the definition of the PLV model selection criterion $\mathcal{G}$ from Section 4.5.1. In practice, we minimise the sum of $\mathcal{G}$ and negative log hyperpriors, the latter are omitted here for simplicity. In this section, we derive the gradient of $\mathcal{G}$ w.r.t. hyperparameters.

Since $Q(\boldsymbol{u}_I)$ depends on the kernel hyperparameters (along with the site parameters $\boldsymbol{b}$, $\boldsymbol{\Pi}$), the correct gradient has to take this dependence into account. In fact, the only assumption we use here is that the site parameters $\boldsymbol{b}$, $\boldsymbol{\Pi}$ are independent of the hyperparameters. Recall that

$$\mathcal{G} = \mathrm{E}_Q \left[-\log P(\boldsymbol{y}|\boldsymbol{u}) \right] + \mathrm{D}[Q(\boldsymbol{u}_I) \,\|\, P(\boldsymbol{u}_I)]$$

with the relative entropy term given by (4.17). For a variation $d\boldsymbol{K}$ we have

$$d\mathrm{D}[Q(\boldsymbol{u}_I) \,\|\, P(\boldsymbol{u}_I)] = \frac{1}{2} \left(d \log |\boldsymbol{M}| + d \operatorname{tr} \boldsymbol{M}^{-1} + d \left\| \boldsymbol{Q}^{-T} \boldsymbol{\beta} \right\|^2 \right).$$

The gradient computations are quite involved, and our strategy is as follows. We aim to write

$$d\mathcal{G} = \operatorname{tr} \boldsymbol{A}^{(1)} (d\boldsymbol{K}_I) + \operatorname{tr} \boldsymbol{A}^{(2)T} (d\boldsymbol{K}_{I,\cdot}) + a^{(3)}$$

where $\boldsymbol{A}^{(1)}$ is symmetric. In the following, we show how to accumulate $\boldsymbol{A}^{(1)}$, $\boldsymbol{A}^{(2)}$. Define

$$\tilde{\boldsymbol{L}} = \boldsymbol{L}\boldsymbol{Q}, \quad \boldsymbol{E} = \tilde{\boldsymbol{L}} \tilde{\boldsymbol{L}}^T = \boldsymbol{K}_I + \boldsymbol{K}_{I,\cdot} \boldsymbol{\Pi} \boldsymbol{K}_{\cdot,I}, \quad \boldsymbol{b}_1 = \tilde{\boldsymbol{L}}^{-T} \boldsymbol{\beta}.$$

Note that

$$d\boldsymbol{E} = d\boldsymbol{K}_I + 2 \operatorname{sym} \left((d\boldsymbol{K}_{I,\cdot}) \boldsymbol{\Pi} \boldsymbol{K}_{\cdot,I} \right),$$

thus

$$\operatorname{tr} \boldsymbol{A}(d\boldsymbol{E}) = \operatorname{tr} \boldsymbol{A}(d\boldsymbol{K}_I) + 2 \operatorname{tr} \left(\boldsymbol{A}\boldsymbol{K}_{I,\cdot}\boldsymbol{\Pi}\right)^T (d\boldsymbol{K}_{I,\cdot}) \tag{C.21}$$

($\boldsymbol{A}$ symmetric) which will frequently be used. We also define $\boldsymbol{B}_1 = \tilde{\boldsymbol{L}}^{-T}\boldsymbol{C} = \boldsymbol{E}^{-1}\boldsymbol{K}_{I,\cdot}$. Note that

$$\boldsymbol{\mu} = \boldsymbol{K}_{\cdot,I}\boldsymbol{b}_1, \quad \boldsymbol{C} = \tilde{\boldsymbol{L}}^{-1}\boldsymbol{K}_{I,\cdot}, \quad \boldsymbol{b}_1 = \boldsymbol{B}_1\boldsymbol{b}, \quad \boldsymbol{E}\boldsymbol{b}_1 = \boldsymbol{K}_{I,\cdot}\boldsymbol{b}. \tag{C.22}$$

We have

$$\left\|\boldsymbol{Q}^{-T}\boldsymbol{\beta}\right\|^2 = \boldsymbol{b}^T\boldsymbol{K}_{I,\cdot}^T\boldsymbol{E}^{-1}\boldsymbol{K}_I\boldsymbol{E}^{-1}\boldsymbol{K}_{I,\cdot}\boldsymbol{b}.$$

If we define

$$\boldsymbol{b}_2 = \boldsymbol{E}^{-1}\boldsymbol{K}_I\boldsymbol{E}^{-1}\boldsymbol{K}_{I,\cdot}\boldsymbol{b} = \boldsymbol{E}^{-1}\boldsymbol{K}_I\boldsymbol{b}_1 = \tilde{\boldsymbol{L}}^{-T}\boldsymbol{Q}^{-1}\boldsymbol{L}^T\boldsymbol{b}_1$$

(using (C.22)), we have

$$d\left\|\boldsymbol{Q}^{-T}\boldsymbol{\beta}\right\|^2 = 2\boldsymbol{b}_2^T(d\boldsymbol{K}_{I,\cdot})\boldsymbol{b} - 2\boldsymbol{b}_1^T(d\boldsymbol{E})\boldsymbol{b}_2 + \boldsymbol{b}_1^T(d\boldsymbol{K}_I)\boldsymbol{b}_1.$$

Now, $\boldsymbol{\Pi}\boldsymbol{K}_{\cdot,I}\boldsymbol{b}_1 = \boldsymbol{\Pi}\boldsymbol{\mu}$ and $\boldsymbol{\Pi}\boldsymbol{K}_{\cdot,I}\boldsymbol{b}_2 = (\boldsymbol{B}_1\boldsymbol{\Pi})^T\boldsymbol{\mu}_I$, and some algebra gives

$$\begin{aligned} d\left\|\boldsymbol{Q}^{-T}\boldsymbol{\beta}\right\|^2 &= \boldsymbol{b}_1^T(d\boldsymbol{K}_I)\left(\boldsymbol{b}_1 - 2\boldsymbol{b}_2\right) + 2\boldsymbol{b}_2^T(d\boldsymbol{K}_{I,\cdot})(\boldsymbol{b} - \boldsymbol{\Pi}\boldsymbol{\mu}) \\ &\quad - 2\boldsymbol{b}_1^T(d\boldsymbol{K}_{I,\cdot})(\boldsymbol{B}_1\boldsymbol{\Pi})^T\boldsymbol{\mu}_I, \end{aligned}$$

thus the accumulation is

$$\boldsymbol{A}^{(1)}{+}= \frac{1}{2}\operatorname{sym}(\boldsymbol{b}_1 - 2\boldsymbol{b}_2)\boldsymbol{b}_1^T, \quad \boldsymbol{A}^{(2)}{+}= \left(\boldsymbol{b}_2(\boldsymbol{b} - \boldsymbol{\Pi}\boldsymbol{\mu})^T - \boldsymbol{b}_1\left(\boldsymbol{\Pi}\boldsymbol{B}_1^T\boldsymbol{\mu}_I\right)^T\right).$$

Next, $\boldsymbol{M} = \boldsymbol{L}^{-1}\boldsymbol{E}\boldsymbol{L}^{-T}$, so that

$$\begin{aligned} d\log|\boldsymbol{M}| &= d\log|\boldsymbol{E}| - d\log|\boldsymbol{K}_I| \\ &= \operatorname{tr}\left(\boldsymbol{E}^{-1} - \boldsymbol{K}_I^{-1}\right)(d\boldsymbol{K}_I) + 2\operatorname{tr}(\boldsymbol{B}_1\boldsymbol{\Pi})^T(d\boldsymbol{K}_{I,\cdot}) \end{aligned}$$

and

$$\boldsymbol{A}^{(1)}{+}= \frac{1}{2}\left(\boldsymbol{E}^{-1} - \boldsymbol{K}_I^{-1}\right), \quad \boldsymbol{A}^{(2)}{+}= \boldsymbol{B}_1\boldsymbol{\Pi}.$$

Next, $\operatorname{tr}\boldsymbol{M}^{-1} = \operatorname{tr}\boldsymbol{E}^{-1}\boldsymbol{K}_I$, thus

$$d\operatorname{tr}\boldsymbol{M}^{-1} = \operatorname{tr}\boldsymbol{E}^{-1}(d\boldsymbol{K}_I) - \operatorname{tr}\boldsymbol{E}^{-1}\boldsymbol{K}_I\boldsymbol{E}^{-1}(d\boldsymbol{E}).$$

Define

$$\boldsymbol{F} = \boldsymbol{E}\boldsymbol{K}_I^{-1}\boldsymbol{E} = \tilde{L}\boldsymbol{Q}^T(\tilde{L}\boldsymbol{Q}^T)^T,$$

then

$$d\operatorname{tr}\boldsymbol{M}^{-1} = \operatorname{tr}\left(\boldsymbol{E}^{-1} - \boldsymbol{F}^{-1}\right)(d\boldsymbol{K}_I) - 2\operatorname{tr}(\boldsymbol{E}^{-1}\boldsymbol{K}_I)^T(d\boldsymbol{K}_{I,\cdot})(\boldsymbol{B}_1\boldsymbol{\Pi})^T$$

Here, $\boldsymbol{E}^{-1}\boldsymbol{K}_I = \tilde{\boldsymbol{L}}^{-T}\boldsymbol{Q}^{-1}\boldsymbol{L}^T$. This leads to the accumulations

$$\boldsymbol{A}^{(1)} += \frac{1}{2}\left(\boldsymbol{E}^{-1} - \boldsymbol{F}^{-1}\right), \quad \boldsymbol{A}^{(2)} += -\boldsymbol{E}^{-1}\boldsymbol{K}_I\boldsymbol{B}_1\boldsymbol{\Pi}.$$

Recall that

$$\mathrm{E}_Q\left[-\log P(\boldsymbol{y}|\boldsymbol{u})\right] = -\sum_{i=1}^{n}\mathrm{E}_Q\left[\log t_i(u_i)\right], \quad Q(u_i) = N(u_i|\mu_i, q_i + l_i^2),$$

where $l_i^2 = \boldsymbol{K}_{i,i} - p_i$ which is 0 for $i \in I$. From (C.9) we know that

$$\mathrm{E}_Q\left[-\log P(\boldsymbol{y}|\boldsymbol{u})\right] = \boldsymbol{d}^T\left(d\boldsymbol{q} + d(l_i^2)_i\right) - \boldsymbol{a}^T d\boldsymbol{\mu}$$

with $\boldsymbol{d}$, $\boldsymbol{a} \in \mathbb{R}^n$ as defined there (see Section C.3.3; replace h_i there by μ_i here, q_i there by $q_i + l_i^2$ here), and we can make use of relation (C.10) once more. Since $\boldsymbol{p} = \operatorname{diag}\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{K}_{I,\cdot}^T\boldsymbol{K}_I^{-1}\boldsymbol{K}_{I,\cdot}$, we have

$$\boldsymbol{d}^T d(l_i^2)_i = \boldsymbol{d}^T(d\operatorname{diag}\boldsymbol{K}) + \boldsymbol{d}^T\operatorname{diag}\left(\boldsymbol{V}^T\boldsymbol{L}^{-1}(d\boldsymbol{K}_I)\boldsymbol{L}^{-T}\boldsymbol{V}\right) - 2\boldsymbol{d}^T\operatorname{diag}\left(\boldsymbol{V}^T\boldsymbol{L}^{-1}(d\boldsymbol{K}_{I,\cdot})\right).$$

Using (C.10), this gives

$$\begin{aligned}\boldsymbol{A}^{(1)} &+= \boldsymbol{L}^{-T}\left(\boldsymbol{V}(\operatorname{diag}\boldsymbol{d})\boldsymbol{V}^T\right)\boldsymbol{L}^{-1}, \quad \boldsymbol{A}^{(2)} += -2\boldsymbol{L}^{-T}\boldsymbol{V}(\operatorname{diag}\boldsymbol{d}),\\ a^{(3)} &+= \boldsymbol{d}^T(d\operatorname{diag}\boldsymbol{K}).\end{aligned}$$

Next, $\boldsymbol{q} = \operatorname{diag}\boldsymbol{C}^T\boldsymbol{C} = \operatorname{diag}\boldsymbol{K}_{I,\cdot}^T\boldsymbol{E}^{-1}\boldsymbol{K}_{I,\cdot}$, thus

$$d\boldsymbol{q} = 2\operatorname{diag}\boldsymbol{B}_1^T(d\boldsymbol{K}_{I,\cdot}) - \operatorname{diag}\boldsymbol{B}_1^T(d\boldsymbol{E})\boldsymbol{B}_1.$$

Using (C.10) and (C.21) and defining

$$\tilde{\boldsymbol{F}} = \boldsymbol{B}_1(\operatorname{diag}\boldsymbol{d})\boldsymbol{B}_1^T,$$

we obtain

$$\boldsymbol{A}^{(1)} += -\tilde{\boldsymbol{F}}, \quad \boldsymbol{A}^{(2)} += \left(2\boldsymbol{B}_1(\operatorname{diag}\boldsymbol{d}) - \tilde{\boldsymbol{F}}\boldsymbol{K}_{I,\cdot}\boldsymbol{\Pi}\right).$$

Finally, $\boldsymbol{\mu} = \boldsymbol{K}_{I,\cdot}^T\boldsymbol{E}^{-1}\boldsymbol{K}_{I,\cdot}\boldsymbol{b}$, thus

$$-\boldsymbol{a}^T d\boldsymbol{\mu} = -\boldsymbol{b}_1^T(d\boldsymbol{K}_{I,\cdot})\boldsymbol{a} + (\boldsymbol{B}_1\boldsymbol{a})^T(d\boldsymbol{E})\boldsymbol{b}_1 - (\boldsymbol{B}_1\boldsymbol{a})^T(d\boldsymbol{K}_{I,\cdot})\boldsymbol{b}$$

and

$$\begin{aligned}
\boldsymbol{A}^{(1)} &+= \operatorname{sym}\boldsymbol{b}_1(\boldsymbol{B}_1\boldsymbol{a})^T, \\
\boldsymbol{A}^{(2)} &+= \left(-\boldsymbol{b}_1\boldsymbol{a}^T - (\boldsymbol{B}_1\boldsymbol{a})\boldsymbol{b}^T + \left(\operatorname{sym}\boldsymbol{b}_1(\boldsymbol{B}_1\boldsymbol{a})^T\right)\boldsymbol{K}_{I,\cdot}\boldsymbol{\Pi}\right).
\end{aligned}$$

How to compute the gradient efficiently, given that in general we expect a fairly large number of hyperparameters? A straightforward accumulation of $\boldsymbol{A}^{(2)}$ is wasteful due to the many different $d \times n$ matrices which are related to each other by $O(n\,d^2)$ back-substitutions. Here, we settle for a method which might be problematic w.r.t. numerical stability. In fact, it is easy to see that

$$\boldsymbol{A}^{(2)} = \boldsymbol{H}^{(0)} + \boldsymbol{H}^{(1)}\boldsymbol{V}\boldsymbol{\Pi} + \boldsymbol{H}^{(2)}\boldsymbol{V}(\operatorname{diag}\boldsymbol{d}),$$

where $\boldsymbol{H}^{(0)}$ can be computed cheaply and $\boldsymbol{H}^{(1)}$, $\boldsymbol{H}^{(2)} \in \mathbb{R}^{d,d}$. Collecting from above and noting that $\boldsymbol{B}_1 = \tilde{\boldsymbol{L}}^{-T}\boldsymbol{Q}^{-1}\boldsymbol{V} = \boldsymbol{E}^{-1}\boldsymbol{L}\boldsymbol{V}$, we have

$$\begin{aligned}
\boldsymbol{H}^{(1)} &= -\boldsymbol{E}^{-1}\boldsymbol{K}_I\tilde{\boldsymbol{L}}^{-T}\boldsymbol{Q}^{-1} + \tilde{\boldsymbol{L}}^{-T}\boldsymbol{Q}^{-1} - \tilde{\boldsymbol{F}}\boldsymbol{L} + \left(\operatorname{sym}\boldsymbol{b}_1(\boldsymbol{B}_1\boldsymbol{a})^T\right)\boldsymbol{L}, \\
\boldsymbol{H}^{(2)} &= -2\boldsymbol{L}^{-T} + 2\tilde{\boldsymbol{L}}^{-T}\boldsymbol{Q}^{-1}, \\
\boldsymbol{H}^{(0)} &= \left(\boldsymbol{b}_2(\boldsymbol{b} - \boldsymbol{\Pi}\boldsymbol{\mu})^T - \boldsymbol{b}_1\left(\boldsymbol{\Pi}\boldsymbol{B}_1^T\boldsymbol{\mu}_I\right)^T\right) - \boldsymbol{b}_1\boldsymbol{a}^T - (\boldsymbol{B}_1\boldsymbol{a})\boldsymbol{b}^T.
\end{aligned}$$

Once the $\boldsymbol{H}^{(i)}$ have been computed, $\boldsymbol{A}^{(2)}$ can be computed overwriting $\boldsymbol{V}$ without the need for a second large buffer. Note that neither $\boldsymbol{C}$ nor $\boldsymbol{B}_1$ have to be computed explicitly although we use them for notational convenience. Thus a sophisticated implementation requires only one $n \times d$ matrix (holding $\boldsymbol{V}$, followed by $\boldsymbol{A}^{(2)}$) since the derivative matrices for $d\boldsymbol{K}_{I,\cdot}$ can be computed on the fly. The gradient computation itself is $O(|\boldsymbol{\alpha}|\,n\,d)$ while the precomputations are $O(n\,d^2)$.

If the likelihood factors t_i have an associated hyperparameter, a variation leads to

$$d\mathcal{G} = -\sum_{i=1}^{n} \mathrm{E}_Q[d\log t_i(u_i)]$$

which again may require numerical quadrature. In the special case where the hyperparameter $\boldsymbol{\alpha}$ is a location parameter in the sense that

$$\frac{d \log t_i(u_i)}{d\alpha} = \frac{d \log t_i(u_i)}{du_i},$$

we can use partial integration to see that

$$\mathrm{E}_Q\left[\frac{d \log t_i(u_i)}{d\alpha}\right] = \mathrm{E}_Q\left[\frac{u_i - \mu_i}{q_i + l_i^2} \log t_i(u_i)\right] = a_i,$$

so that $d\mathcal{G} = -\boldsymbol{a}^T \mathbf{1} d\alpha$. Note that bias parameters in classification noise models such as the probit or logit one are location parameters.

Bibliography

[1] P. Abrahamsen. A review of Gaussian random fields and correlation functions. Technical report, Norwegian Computing Centre, 1997.

[2] R. J. Adler. *Geometry of Random Fields*. John Wiley & Sons, 1981.

[3] M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[4] M. Anthony and N. Biggs. *Computational Learning Theory*. Cambridge University Press, 1997.

[5] M. Anthony and J. Shawe-Taylor. A result of Vapnik with applications. *Discrete Applied Mathematics*, 47(2):207–217, 1993.

[6] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68(3):337–404, 1950.

[7] H. Attias. A variational Bayesian framework for graphical models. In Solla et al. [185], pages 209–215.

[8] F. Bach and M. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48, 2002.

[9] C. T. H. Baker. *The Numerical Treatment of Integral Equations*. Clarendon Press, Oxford, 1977.

[10] Y. Bar-Shalom and X. Li. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House, 1993.

[11] D. Barber and C. Bishop. Ensemble learning for multi-layer networks. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 395–401. MIT Press, 1998.

[12] P. Bartlett, S. Bocheron, and G. Lugosi. Model selection and error estimation. In N. Cesa-Bianchi and S. Goldman, editors, *Conference on Computational Learning Theory 13*, pages 286–297. Morgan Kaufmann, 2000.

[13] S. Becker, S. Thrun, and K. Obermayer, editors. *Advances in Neural Information Processing Systems 15*. MIT Press, 2003. To appear.

[14] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 2nd edition, 1985.

[15] José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. John Wiley & Sons, 1st edition, 1994.

[16] P. Billingsley. *Probability and Measure*. John Wiley & Sons, 3rd edition, 1995.

[17] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1st edition, 1995.

[18] C. Bishop and B. Frey, editors. *Workshop on Artificial Intelligence and Statistics 9*, 2003. Electronic Proceedings (ISBN 0-9727358-0-1).

[19] C. Bishop and J. Winn. Structured variational distributions in VIBES. In Bishop and Frey [18], pages 244–251. Electronic Proceedings (ISBN 0-9727358-0-1).

[20] S. Boucheron, G. Lugosi, and P. Massart. A sharp concentration inequality with applications. *Random Structures and Algorithms*, 16:277–292, 2000.

[21] O. Bousquet. A Bennett concentration inequality and its application to suprema of empirical processes. *C. R. Acad. Sci. Paris*, 334:495–500, 2002.

[22] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[23] G. Box and G. Tiao. *Bayesian Inference in Statistical Analysis*. John Wiley & Sons, 1992.

[24] S. Boyd and L. Vandenberghe. *Convex Optimization*. 2002. Available online at `www.stanford.edu/~boyd/cvxbook.html`.

[25] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In G. Cooper and S. Moral, editors, *Uncertainty in Artificial Intelligence 14*. Morgan Kaufmann, 1998.

[26] N. Brenner. Matrix transposition in place. *Communications of the ACM*, 16(11):692–694, 1973. CACM Algorithm 467.

[27] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning. In Leen et al. [101], pages 409–415.

[28] H. Chernoff. A measure of asymptotic efficiency of tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

[29] K. L. Chung. *A Course in Probability Theory*. Academic Press, 2nd edition, 1974.

[30] B. S. Clarke and A. R. Barron. Information-theoretic asymptotics of Bayes methods. *IEEE Transactions on Information Theory*, 36(3):453–471, 1990.

[31] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[32] C. Cortes, P. Haffner, and M. Mohri. Rational kernels. In Becker et al. [13]. To appear.

[33] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[34] Thomas Cover and Joy Thomas. *Elements of Information Theory*. Series in Telecommunications. John Wiley & Sons, 1st edition, 1991.

[35] H. Cramér. Sur un nouveau théorème limite de la théorie des probabilités. *Actualités Sci. Indust.*, 736, 1938.

[36] N. Cressie. *Statistics for Spatial Data*. John Wiley & Sons, 2nd edition, 1993.

[37] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel Based Methods*. Cambridge University Press, 2000.

[38] Lehel Csató. *Gaussian Processes — Iterative Sparse Approximations*. PhD thesis, Aston University, Birmingham, UK, March 2002.

[39] Lehel Csató and Manfred Opper. Sparse representations of Gaussian processes. In Leen et al. [101], pages 444–450.

[40] Lehel Csató and Manfred Opper. Sparse online Gaussian processes. *Neural Computation*, 14:641–668, 2002.

[41] Lehel Csató, Manfred Opper, and Ole Winther. TAP Gibbs free energy, belief propagation and sparsity. In Dietterich et al. [50], pages 657–663.

[42] I. Csiszár and J. Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, 1981.

[43] I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. In et. al. E. F. Dedewicz, editor, *Statistics and Decisions*, pages 205–237. Oldenburg Verlag, Munich, 1984.

[44] G. Cybenko and M. Berry. Hyperbolic Householder algorithms for factoring structured matrices. *SIAM J. Matrix Anal. Appl.*, 11:499–520, 1990.

[45] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[46] A. Darwiche and N. Friedman, editors. *Uncertainty in Artificial Intelligence 18*. Morgan Kaufmann, 2002.

[47] P. Davis and P. Rabinovitz. *Methods of Numerical Integration*. Academic Press, 1984.

[48] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Roy. Stat. Soc. B*, 39:1–38, 1977.

[49] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Applications of Mathematics: Stochastic Modelling and Applied Probability. Springer, 1st edition, 1996.

[50] T. Dietterich, S. Becker, and Z. Ghahramani, editors. *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[51] J. Dongarra, J. Du Croz, S. Hammarling, and R. Hanson. An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 14:1–17, 1988.

[52] A. Faul and M. Tipping. Analysis of sparse Bayesian learning. In Dietterich et al. [50], pages 383–389.

[53] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972.

[54] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

[55] Roger Fletcher. *Practical Methods of Optimization: Unconstrained Optimization*, volume 1. John Wiley & Sons, 1980.

[56] Roger Fletcher. *Practical Methods of Optimization: Constrained Optimization*, volume 2. John Wiley & Sons, 1989.

[57] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In L. Saitta, editor, *International Conference on Machine Learning 13*, pages 148–156. Morgan Kaufmann, 1996.

[58] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

[59] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Stanford University, 1998.

[60] N. Friedman and I. Nachman. Gaussian process networks. In C. Boutilier and M. Goldszmidt, editors, *Uncertainty in Artificial Intelligence 16*, pages 211–219. Morgan Kaufmann, 2000.

[61] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. In *Foundations of Computer Science 39*, pages 370–378, 1998.

[62] Z. Ghahramani and M. Beal. Propagation algorithms for variational Bayesian learning. In Leen et al. [101], pages 507–513.

[63] Mark N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

[64] W. Gilks, S. Richardson, and D. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1st edition, 1996.

[65] A. Girard, C. Rasmussen, and R. Murray-Smith. Gaussian process priors with uncertain inputs — application to multiple-step ahead time series forecasting. In Becker et al. [13]. To appear.

[66] P.J. Green and Bernhard Silverman. *Nonparametric Regression and Generalized Linear Models*. Monographs on Statistics and Probability. Chapman & Hall, 1994.

[67] Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*. Oxford University Press, 3rd edition, 2001.

[68] L. Györfi, editor. *Principles of Nonparametric Learning*. Springer, 2002.

[69] David Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, July 1999. See `http://www.cse.ucsc.edu/~haussler/pubs.html`.

[70] David Haussler, Michael Kearns, and Robert Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. *Machine Learning*, 14:83–113, 1994.

[71] David Haussler and Manfred Opper. Mutual information, metric entropy and cumulative relative entropy risk. *Annals of Statistics*, 25(6):2451–2492, 1997.

[72] Ralf Herbrich. *Learning Kernel Classifiers*. MIT Press, 1st edition, 2001.

[73] Ralf Herbrich and Thore Graepel. A PAC-Bayesian margin bound for linear classifiers: Why SVMs work. In Leen et al. [101], pages 224–230.

[74] Tom Heskes and Onno Zoeter. Expectation propagation for approximate inference in dynamic Bayesian networks. In Darwiche and Friedman [46].

[75] G. E. Hinton and R. M. Neal. A new view on the EM algorithm that justifies incremental and other variants. In Jordan [85].

[76] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*. Number 305 in Grundlehren der mathematischen Wissenschaften. Springer, 1993.

[77] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*. Number 306 in Grundlehren der mathematischen Wissenschaften. Springer, 1993.

[78] Roger Horn and Charles Johnson. *Matrix Analysis*. Cambridge University Press, 1st edition, 1985.

[79] Shunsuke Ihara. *Information Theory for Continuous Systems*. World Scientific, 1st edition, 1993.

[80] T. Jaakkola. *Variational Methods for Inference and Estimation in Graphical Models*. PhD thesis, Massachusetts Institute of Technology, 1997.

[81] T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In Kearns et al. [87], pages 487–493.

[82] Tommi Jaakkola and David Haussler. Probabilistic kernel regression models. In D. Heckerman and J. Whittaker, editors, *Workshop on Artificial Intelligence and Statistics 7*. Morgan Kaufmann, 1999.

[83] Tommi Jaakkola, Marina Meila, and Tony Jebara. Maximum entropy discrimination. In Solla et al. [185], pages 470–476.

[84] M. Jordan. An introduction to probabilistic graphical models. In preparation, 2003.

[85] M. I. Jordan, editor. *Learning in Graphical Models*. Kluwer, 1997.

[86] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[87] M. Kearns, S. Solla, and D. Cohn, editors. *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[88] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[89] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

[90] A. N. Kolmogorov. *Foundations of the Theory of Probability*. Chelsea, New York, 2nd edition, 1933. Trans. N. Morrison (1956).

[91] V. Koltchinskii and D. Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. *Annals of Statistics*, 30(1):1–50, 2002.

[92] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In C. Sammut and A. Hofmann, editors, *International Conference on Machine Learning 19*. Morgan Kaufmann, 2002.

[93] D. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metallurgical and Mining Society of South Africa*, 52:119–139, 1951.

[94] J. Langford and J. Shawe-Taylor. PAC-Bayes and margin. In Becker et al. [13]. To appear.

[95] John Langford and Rich Caruana. (Not) bounding the true error. In Dietterich et al. [50], pages 809–816.

[96] P. Langley, editor. *International Conference on Machine Learning 17*. Morgan Kaufmann, 2000.

[97] Neil D. Lawrence and Ralf Herbrich. A sparse Bayesian compression scheme - the informative vector machine. Presented at NIPS 2001 Workshop on Kernel Methods, 2001.

[98] Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Becker et al. [13]. See `www.dai.ed.ac.uk/~seeger/papers.html`.

[99] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.

[100] M. Ledoux and M. Talagrand. *Probability in Banach Spaces*. Springer, 1991.

[101] T. Leen, T. Dietterich, and V. Tresp, editors. *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[102] U. Lerner. *Hybrid Bayesian Networks*. PhD thesis, Stanford University, 2002.

[103] Nick Littlestone and Manfred Warmuth. Relating data compression and learnability. Technical report, University of California, Santa Cruz, 1986.

[104] Jun Liu. Markov chain Monte Carlo and related topics. Technical report, Department of Statistics, Stanford University, 1999.

[105] G. Lugosi. Pattern classification and learning theory. In Györfi [68].

[106] G. Lugosi. Concentration-of-measure inequalities. Technical report, 2003. Presented at Summer School Machine Learning, ANU Canberra.

[107] Z. Luo and G. Wahba. Hybrid adaptive splines. *Journal of the American Statistical Association*, 92:107–116, 1997.

[108] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1991.

[109] D. MacKay. Bayesian non-linear modeling for the energy prediction competition. In *ASHRAE Transactions*, volume 100, pages 1053–1062, 1994.

[110] D. MacKay. Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks. *Network — Computation in Neural Systems*, 6(3):469–505, 1995.

[111] D. MacKay. Introduction to Gaussian processes. Technical report, Cambridge University, UK, 1997. See `http://wol.ra.phy.cam.ac.uk/mackay/README.html`.

[112] G. Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–1266, 1963.

[113] G. Matheron. The intrinsic random functions and their applications. *Journal for Applied Probability*, 5:439–468, 1973.

[114] D. McAllester and L. Ortiz. Concentration inequalities for the missing mass and for histogram rule error. In Becker et al. [13]. To appear.

[115] David McAllester. PAC-Bayesian model averaging. In *Conference on Computational Learning Theory 12*, pages 164–170, 1999.

[116] David McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[117] David McAllester. PAC-Bayesian stochastic model selection. To appear in Machine Learning. See `www.autoreason.com`, 2002.

[118] P. McCullach and J.A. Nelder. *Generalized Linear Models*. Number 37 in Monographs on Statistics and Applied Probability. Chapman & Hall, 1st edition, 1983.

[119] R. McEliece, D. MacKay, and J.-F. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998.

[120] J. McNamee and F. Stenger. Construction of fully symmetric numerical integration formulas. *Numerische Mathematik*, 10:327–344, 1967.

[121] Ron Meir and Tong Zhang. Data-dependent bounds for Bayesian mixture methods. In Becker et al. [13]. See `citeseer.nj.nec.com/536920.html`.

[122] T. Minka and R. Picard. Learning how to learn is learning with point sets. Unpublished manuscript. Available at `http://wwwwhite.media.mit.edu/~tpminka/papers/learning.html`., 1997.

[123] Thomas Minka. The EP energy function and minimization schemes. See `www.stat.cmu.edu/~minka/papers/learning.html`, August 2001.

[124] Thomas Minka. Expectation propagation for approximate Bayesian inference. In J. Breese and D. Koller, editors, *Uncertainty in Artificial Intelligence 17*. Morgan Kaufmann, 2001.

[125] Thomas Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, January 2001.

[126] E. H. Moore. On properly positive hermitian matrices. *Bull. Amer. Math. Soc.*, 23:59, 1916.

[127] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2003.

[128] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In K. Laskey and H. Prade, editors, *Uncertainty in Artificial Intelligence 15*, pages 467–475. Morgan Kaufmann, 1999.

[129] B. Natajaran. Sparse approximate solutions to linear systems. *SIAM Journal of Computing*, 24(2):227–234, 1995.

[130] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, University of Toronto, 1993.

[131] R. M. Neal. *Bayesian Learning for Neural Networks*. Number 118 in Lecture Notes in Statistics. Springer, 1996.

[132] Radford M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian classification and regression. Technical Report 9702, Department of Statistics, University of Toronto, January 1997.

[133] J. Nelder and R. Wedderburn. Generalized linear models. *Journal of Roy. Stat. Soc. B*, 135:370–384, 1972.

[134] D. Nychka, G. Wahba, S. Goldfarb, and T. Pugh. Cross-validated spline methods for the estimation of three dimensional tumor size distributions from observations on two dimensional cross sections. *Journal of the American Statistical Association*, 79:832–846, 1984.

[135] A. O'Hagan. Curve fitting and optimal design. *Journal of Roy. Stat. Soc. B*, 40(1):1–42, 1978.

[136] A. O'Hagan. Some Bayesian numerical analysis. In J. Bernardo, J. Berger, A. Dawid, and A. Smith, editors, *Bayesian Statistics 4*, pages 345–363. Oxford University Press, 1992.

[137] M. Opper and O. Winther. Gaussian process classification and SVM: Mean field results and leave-one-out estimator. In Smola et al. [179].

[138] Manfred Opper. A Bayesian approach to on-line learning. In D. Saad, editor, *On-Line Learning in Neural Networks*. Cambridge University Press, 1998.

[139] Manfred Opper and Ole Winther. Gaussian processes for classification: Mean field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

[140] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Smola et al. [179].

[141] J. Platt, C. Burges, S. Swenson, C. Weare, and A. Zheng. Learning a Gaussian process prior for automatically generating music playlists. In Dietterich et al. [50], pages 1425–1432.

[142] John C. Platt. Fast training of support vector machines using sequential minimal optimization. In Schölkopf et al. [160], pages 185–208.

[143] M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, 4(2):305–318, 1993.

[144] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78(9):1481–1497, 1990.

[145] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[146] C. E. Rasmussen. *Evaluation of Gaussian Processes and Other Methods for Nonlinear Regression*. PhD thesis, University of Toronto, 1996.

[147] C. E. Rasmussen. The infinite Gaussian mixture model. In Solla et al. [185], pages 554–560.

[148] C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In Dietterich et al. [50], pages 881–888.

[149] R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26:195–239, 1984.

[150] B. D. Ripley. *Stochastic Simulation*. John Wiley & Sons, 1987.

[151] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Texts in Statistics. Springer, 1st edition, 1999.

[152] R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[153] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[154] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[155] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[156] P. Rujan. A fast method for calculating the perceptron with maximal stability. *Journal de Physique I*, 3:277–290, 1993.

[157] Y. Saad. *Iterative Methods for Sparse Linear Systems*. International Thomson Publishing, 1st edition, 1996.

[158] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

[159] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

[160] B. Schölkopf, C. Burges, and A. Smola, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1998.

[161] Bernhard Schölkopf and Alexander Smola. *Learning with Kernels*. MIT Press, 1st edition, 2002.

[162] I. J. Schönberg. Metric spaces and completely monotone functions. In *Proc. Nat. Acad. Sci.*, volume 39, pages 811–841, 1938.

[163] I. J. Schönberg. Spline functions and the problem of graduation. *Annals of Mathematicals*, 52:947–950, 1964.

[164] M. Seeger. Covariance kernels from Bayesian generative models. In Dietterich et al. [50], pages 905–912.

[165] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In Bishop and Frey [18], pages 205–212. Electronic Proceedings (ISBN 0-9727358-0-1).

[166] Matthias Seeger. Bayesian methods for support vector machines and Gaussian processes. Master's thesis, University of Karlsruhe, Germany, 1999. See `http://www.dai.ed.ac.uk/~seeger/papers.html`.

[167] Matthias Seeger. Annealed expectation-maximization by entropy projection. Technical report, Institute for ANC, Edinburgh, UK, 2000. See `http://www.dai.ed.ac.uk/~seeger/papers.html`.

[168] Matthias Seeger. Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers. In Solla et al. [185], pages 603–609.

[169] Matthias Seeger. Covariance kernels from Bayesian generative models. Technical report, Institute for ANC, Edinburgh, UK, 2000. See `http://www.dai.ed.ac.uk/~seeger/papers.html`.

[170] Matthias Seeger. PAC-Bayesian generalization error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, October 2002.

[171] Matthias Seeger. PAC-Bayesian generalization error bounds for Gaussian process classification. Technical Report EDI-INF-RR-0094, Division of Informatics, University of Edinburgh, 2002. See `www.dai.ed.ac.uk/~seeger/papers.html`.

[172] Matthias Seeger, John Langford, and Nimrod Megiddo. An improved predictive accuracy bound for averaging classifiers. In C. E. Brodley and A. P. Danyluk, editors, *International Conference on Machine Learning 18*, pages 290–297. Morgan Kaufmann, 2001.

[173] Matthias Seeger, Neil D. Lawrence, and Ralf Herbrich. Sparse Bayesian learning: The informative vector machine. Technical report, Department of Computer Science, Sheffield, UK, 2002. See `www.dcs.shef.ac.uk/~neil/papers/`.

[174] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Conference on Computational Learning Theory 5*, pages 287–294. Morgan Kaufmann, 1992.

[175] J. Shawe-Taylor and C. Williams. The stability of kernel principal components analysis and its relation to the process eigenspectrum. In Becker et al. [13]. To appear.

[176] J. Shawe-Taylor and R. Williamson. A PAC analysis of a Bayesian estimator. In *Conference on Computational Learning Theory 10*, pages 2–9, 1997.

[177] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5):1926–1940, 1998.

[178] J. Skilling. Bayesian numerical analysis. In John Skilling, editor, *Maximum Entropy and Bayesian Methods*. Cambridge University Press, 1989.

[179] A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors. *Advances in Large Margin Classifiers*. MIT Press, 1999.

[180] A. Smola, Z. Óvári, and R. Williamson. Regularization with dot-product kernels. In Leen et al. [101], pages 308–314.

[181] A. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In Langley [96], pages 911–918.

[182] A. Smola, B. Schölkopf, and K.-R. Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11:637–649, 1998.

[183] Alex Smola and Peter Bartlett. Sparse greedy Gaussian process regression. In Leen et al. [101], pages 619–625.

[184] E. Solak, R. Murray-Smith, W. Leithead, and C. Rasmussen. Derivative observations in Gaussian process models of dynamic systems. In Becker et al. [13]. To appear.

[185] S. Solla, T. Leen, and K.-R. Müller, editors. *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[186] Peter Sollich. Learning curves for Gaussian processes. In Kearns et al. [87], pages 344–350.

[187] Peter Sollich. Probabilistic methods for support vector machines. In Solla et al. [185], pages 349–355.

[188] D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. BUGS: Bayesian inference using Gibbs sampling. Technical report, MRC Biostatistics Unit, Cambridge University, 1995.

[189] M. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer, 1999.

[190] Y. Teh and M. Welling. On improving the efficiency of the iterative proportional fitting procedure. In Bishop and Frey [18], pages 213–220. Electronic Proceedings (ISBN 0-9727358-0-1).

[191] J. Tenenbaum, A. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[192] Michael Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[193] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2001.

[194] Volker Tresp. A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741, 2000.

[195] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[196] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Applications*, 16(2):264–280, 1971.

[197] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1st edition, 1998.

[198] F. Vivarelli and C. K. I. Williams. Discovering hidden features with Gaussian process regression. In Kearns et al. [87].

[199] A. van der Waart and J. Wellner. *Weak Convergence and Empirical Processes*. Springer, 1996.

[200] Grace Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series. SIAM Society for Industrial and Applied Mathematics, 1990.

[201] Grace Wahba. Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV. In Schölkopf et al. [160], pages 69–88.

[202] M. Wainwright, E. Sudderth, and A. Willsky. Tree-based modeling and estimation of Gaussian processes on graphs with cycles. In Leen et al. [101], pages 661–667.

[203] M. J. Wainwright. *Stochastic Processes on Graphs with Cycles*. PhD thesis, Massachusetts Institute of Technology, January 2002.

[204] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. A new class of upper bounds on the log partition function. In Darwiche and Friedman [46].

[205] S. R. Waterhouse and A. J. Robinson. Classification using hierarchical mixtures of experts. In *IEEE Workshop on Neural Networks for Signal Processing 4*, pages 177–186, 1994.

[206] Y. Weiss and W. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. In Solla et al. [185], pages 673–679.

[207] C. Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.

[208] Christopher Williams and Matthias Seeger. The effect of the input density distribution on kernel-based classifiers. In Langley [96], pages 1159–1166.

[209] Christopher K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan [85].

[210] Christopher K. I. Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[211] Christopher K. I. Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In Leen et al. [101], pages 682–688.

[212] S. Wright. Modified Cholesky factorizations in interior-point algorithmsfor linear programming. *SIAM Journal on Optimization*, 9(4):1159–1191, 1999.

[213] L. Xu and M. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, 1996.

[214] A. Yaglom. *Correlation Theory of Stationary and Related Random Functions*, volume I. Springer, 1987.

[215] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In Leen et al. [101], pages 689–695.

[216] H. Zhu, C. K. I. Williams, R. Rohwer, and M. Morciniec. Gaussian regression and optimal finite dimensional linear models. In C. Bishop, editor, *Neural Networks and Machine Learning*, volume 168 of *NATO ASI series*. Springer, 1998.