Learning with Uncertainty – Gaussian Processes and Relevance Vector Machines

Joaquin Quiñonero Candela

Kongens Lyngby 2004
IMM-PHD-2004-135

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD: ISSN 0909-3192

Synopsis

Denne afhandling omhandler Gaussiske Processer (GP) og Relevans Vektor Maskiner (RVM), som begge er specialtilfælde af probabilistiske lineære modeller. Vi anskuer begge modeller fra en Bayesiansk synsvinkel og er tvunget til at benytte approximativ Bayesiansk behandling af indlæring af to grunde. Først fordi den fulde Bayesianske løsning ikke er analytisk mulig og fordi vi af princip ikke vil benytte metoder baseret på sampling. For det andet, som understøtter kravet om ikke at bruge sampling er ønsket om beregningsmæssigt effektive modeller. Beregningmæssige besparelser opnås ved hjælp af udtyndning: udtyndede modeller har et stort antal parametre sat til nul. For RVM, som vi behandler i kapitel 2 vises at det er det specifikke valg af Bayesiank approximation som resulterer i udtynding. Probabilistiske modeller har den vigtige egenskab der kan beregnes prediktive fordelinger istedet for punktformige prediktioner. Det vises også at de resulterende undtyndede probabilistiske modeller implicerer ikke-intuitive a priori fordelinger over funktioner, og ultimativt utilstrækkelige prediktive varianser; modellerne er mere sikre på sine prediktioner jo længere væk man kommer fra træningsdata. Vi foreslår RVM*, en modificeret RVM som producerer signifikant bedre prediktive usikkerheder. RVM er en speciel klasse af GP, de sidstnævnte giver bedre resultater og er ikke-udtyndede ikke-parametriske modeller. For komplethedens skyld, i kapitel 3 studerer vi en specifik familie af approksimationer, Reduceret Rank Gaussiske Processer (RRGP), som tager form af endelige udviddede lineære modeller. Vi viser at Gaussiaske Processer generelt er ækvivalente med uendelige udviddede lineære modeller. Vi viser også at RRGP, ligesom RVM lider under utilstraekkelige prediktive varianser. Dette problem løses ved at modificere den klassiske RRGP metode analogt til RVM*. I den sidste del af afhandlingen bevæger vi os til problemstillinger med usikre input. Disse er indtil nu antaget deterministiske, hvilket er det

gængse. Her udleder vi ligninger for prediktioner ved stokastiske input med GP og RVM og bruger dem til at propagere usikkerheder rekursivt multiple skridt frem for tidsserie prediktioner. Det tilader os at beregne fornuftige usikkerheder ved rekursiv prediktion k skridt frem i tilfælde hvor standardmetoder som ignorerer den akkumulerende usikkerhed vildt overestimerer deres konfidens. Til slut undersøges et meget sværere problem: træning med usikre inputs. Vi undersøger den fulde Bayesianske løsning som involverer et uløseligt integral. Vi foreslår to preliminære løsninger. Den første forsøger at "gætte" de ukendte "rigtige" inputs, og kræver finjusteret optimering for at undgå overfitning. Den kræver også a priori viden af output støjen, hvilket er en begrænsning. Den anden metode beror på sampling fra inputenes a posterior fordeling og optimisering af hyperparametrene. Sampling har som bivirkning en kraftig forøgelse af beregningsarbejdet, som igen er en begrænsning. Men, success på legetøjseksempler er opmuntrende og skulle stimulere fremtidig forskning.

Summary

This thesis is concerned with Gaussian Processes (GPs) and Relevance Vector Machines (RVMs), both of which are particular instances of probabilistic linear models. We look at both models from a Bayesian perspective, and are forced to adopt an approximate Bayesian treatment to learning for two reasons. The first reason is the analytical intractability of the full Bayesian treatment and the fact that we in principle do not want to resort to sampling methods. The second reason, which incidentally justifies our not wanting to sample, is that we are interested in computationally efficient models. Computational efficiency is obtained through sparseness: sparse linear models have a significant number of their weights set to zero. For the RVM, which we treat in Chap. 2, we show that it is precisely the particular choice of Bayesian approximation that enforces sparseness. Probabilistic models have the important property of producing predictive distributions instead of point predictions. We also show that the resulting sparse probabilistic model implies counterintuitive priors over functions, and ultimately inappropriate predictive variances; the model is more certain about its predictions, the further away from the training data. We propose the RVM*, a modified RVM that provides significantly better predictive uncertainties. RVMs happen to be a particular case of GPs, the latter having superior performance and being non-sparse non-parametric models. For completeness, in Chap. 3 we study a particular family of approximations to Gaussian Processes, Reduced Rank Gaussian Processes (RRGPs), which take the form of finite extended linear models; we show that GPs are in general equivalent to infinite extended linear models. We also show that RRGPs result in degenerate GPs, which suffer, like RVMs, of inappropriate predictive variances. We solve this problem in by proposing a modification of the classic RRGP approach, in the same guise as the RVM*. In the last part of this thesis we move on to the

problem of uncertainty in the inputs. Indeed, these were until now considered deterministic, as it is common use. We derive the equations for predicting at an uncertain input with GPs and RVMs, and use this to propagate the uncertainty in recursive multi-step ahead time-series predictions. This allows us to obtain sensible predictive uncertainties when recursively predicting k-steps ahead, while standard approaches that ignore the accumulated uncertainty are way overconfident. Finally we explore a much harder problem: that of training with uncertain inputs. We explore approximating the full Bayesian treatment, which implies an analytically intractable integral. We propose two preliminary approaches. The first one tries to "guess" the unknown "true" inputs, and requires careful optimisation to avoid over-fitting. It also requires prior knowledge of the output noise, which is limiting. The second approach consists in sampling from the inputs posterior, and optimising the hyperparameters. Sampling has the effect of severely incrementing the computational cost, which again is limiting. However, the success in toy experiments is exciting, and should motivate future research.

Preface

This thesis was prepared partly at Informatics Mathematical Modelling, at the Technical University of Denmark, partly at the Department of Computer Science, at the University of Toronto, and partly at the Max Planck Institute for Biological Cybernetics, in Tübingen, Germany, in partial fulfilment of the requirements for acquiring the Ph.D. degree in engineering.

The thesis deals with probabilistic extended linear models for regression, under approximate Bayesian learning schemes. In particular the Relevance Vector Machine and Gaussian Processes are studied. One focus is guaranteeing computational efficiency while at the same implying appropriate priors over functions. The other focus is to deal with uncertainty in the inputs, both at test and at training time.

The thesis consists of a summary report and a collection of five research papers written during the period 2001–2004, and elsewhere published.

Tübingen, May 2004

Joaquin Quiñonero Candela

Papers included in the thesis

[B] Joaquin Quiñonero-Candela and Lars Kai Hansen. (2002). Time Series Prediction Based on the Relevance Vector Machine with Adaptive Kernels. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2002*, volume 1, pages 985-988, Piscataway, New Jersey, IEEE. This paper was awarded an oral presentation.

[C] Joaquin Quiñonero-Candela and Ole Winther. (2003). Incremental Gaussian Processes. In Becker, S., Thrun, S., and Obermayer, L., editors, *Advances in Neural Information Processing Systems 15*, pages 1001–1008, Cambridge, Massachussetts. MIT Press.

[D] Agathe Girard, Carl Edward Rasmussen, Joaquin Quiñonero-Candela and Roderick Murray-Smith. (2003). Gaussian Process with Uncertain Inputs - Application to Multiple-Step Ahead Time-Series Forecasting. In Becker, S., Thrun, S., and Obermayer, L., editors, *Advances in Neural Information Processing Systems 15*, pages 529–536, Cambridge, Massachussetts. MIT Press. This paper was awarded an oral presentation.

[E] Joaquin Quiñonero-Candela and Agathe Girard and Jan Larsen and Carl E. Rasmussen. (2003). Propagation of Uncertainty in Bayesian Kernels Models - Application to Multiple-Step Ahead Forecasting. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 701–704, Piscataway, New Jersey, IEEE. This paper was awarded an oral presentation.

[F] Fabian Sinz, Joaquin Quiñonero-Candela, Goekhan Bakır, Carl Edward Rasmussen and Matthias O. Franz. (2004). Learning Depth from Stereo. To appear in *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM) Pattern Recognition Symposium 26*, Heidelberg, Germany. Springer.

Acknowledgements

Before anything, I would like to thank my advisors, Lars Kai Hansen and Carl Edward Rasmussen, for having given me the opportunity to do this PhD. I have received great support and at the same time been given large freedom to conduct my research. It is thanks to them that I have been able to spend a part of these three years at University of Toronto, and another part at the Max Planck Institute in Tübingen. These visits to other groups have had the most beneficial effects for my research, and for my career as a scientist. I have also very much enjoyed the work together, which has inspired most of what is contained in this thesis.

My PhD has been funded by the *Multi-Agent Control* (MAC) Research and Training Network of the European Commission, coordinated by Roderick Murray-Smith to whom I feel indebted. Being in MAC allowed me to meet other young researchers, two of which I would like to highlight: Kai Wulff, with whom I have had very interesting discussions, and Agathe Girard with whom I have had the pleasure to collaborate. Their friendship and support made my PhD an easier task.

The two years I spent in Denmark have been wonderful, to a great extent thanks to the many fantastic people at the ISP group. I would like to begin with Ulla Nørhave, whose help in many diverse practical matters has been invaluable. It has been a pleasure collaborating with Jan Larsen and with Ole Winther, from whom I have learned much. I have had lots of fun with my fellow PhD students at ISP, especially with Thomas Fabricius (and Anita), Thomas Kolenda (and Sanne), Siggi, Anna Szymkowiak, and with Tue Lehn-Schiøler, Anders Meng, Rasmus Elsborg Madsen, and Niels Pontoppidan. Mange tak!

I am very grateful to Sam Roweis, for receiving me as a visitor for six months at the Department of Computer Science, at the University of Toronto. Working with him was extremely inspiring. In Toronto I also had met Miguel Ángel Carreira Perpiñán, Max Welling, Jakob Verbeek and Jacob Goldberger, with whom I had very fruitful discussions, and who contributed to making my stay in Toronto very pleasant.

I would like give warm thanks to Bernhard Schölkopf for taking me in his group as a visiting PhD student, and now as a postdoc, at the Max Planck Institute in Tübingen. I have met here an amazing group of people, from whom there is a lot to learn, and with whom leisure time is great fun! I would like to thank Olivier Bousquet and Arthur Gretton for proofreading part of this thesis, Olivier Chapelle and Alex Zien (my office mates), Malte Kuss and Lehel Csató for great discussions. To them and to the rest of the group I am indebted for the great atmosphere they contribute to create.

All this wouldn't have been possible in the first place without the immense support from my parents and sister, to whom I am more than grateful. Last, but most important, I would like to mention the loving support and extreme patience of my wife Ines Koch: I wouldn't have made it without her help. To all my family I would like to dedicate this thesis.

Contents

CHAPTER 1

Introduction

In this thesis we address the univariate regression problem. This is a supervised learning problem, where we are given a training dataset composed of pairs of inputs (in an Euclidean space of some dimension) and outputs or targets (in a unidimensional Euclidean space). We study two models for regression: the Relevance Vector Machine (RVM) and the Gaussian Process (GP) model. Both are instances of probabilistic extended linear models, that perform linear regression on the (possibly non-linearly transformed) inputs. For both models we will consider approximations to a full Bayesian treatment, that yield sparse solutions in the case of the RVM, and that allow for computationally efficient approaches in the case of GPs. These approximations to the full Bayesian treatment come at the cost of poor priors over functions, which result in inappropriate and counter-intuitive predictive variances. Since the predictive variances are a key outcome of probabilistic models, we propose ways of significantly improving them, while preserving computational efficiency. We also address the case where uncertainty arises in the inputs, and we derive the equations for predicting at uncertain test inputs with GPs and RVMs. We also discuss ways of solving a harder task: that of training GPs with uncertain inputs. Below we provide a more detailed description of the main three parts of this thesis: the study of RVMs, the study of computationally efficient approximations to GP, and the study of predicting and training on uncertain inputs.

In the Bayesian perspective, instead of learning point estimates of the model parameters, one considers them as random variables and infers posterior distributions over them. These posterior distributions incorporate the evidence brought up by the training data, and the prior assumptions on the parameters expressed by means of prior distributions over them. In Chap. 2 we describe the extended linear models: these map the inputs (or non-linear transformations of them) into function values as linear combinations under some weights. We discuss Bayesian extended linear models, with prior distributions on the weights, and establish their relation to regularised linear models, as widely used in classical data fitting. Regularisation has the effect of guaranteeing stability and enforcing smoothness through forcing the weights to remain small. We then move on to discussing the Relevance Vector Machine (RVM), recently introduced by Tipping (2001). The RVM is an approximate Bayesian treatment of extended linear models, which happens to enforce sparse solutions. Sparseness means that a significant number of the weights are zero (or effectively zero), which has the consequence of producing compact, computationally efficient models, which in addition are simple and therefore produce smooth functions. We explain how sparseness arises as a result of a particular approximation to a full Bayesian treatment. Indeed, full Bayesian methods would not lend themselves to sparseness, since there is always posterior mass on non-sparse solutions. We apply the RVM to time-series predictions, and show that much can be gained from adapting the basis functions that are used to non-linearly map the inputs to the space on which linear regression is performed. Training RVMs remains computationally expensive -cubic in the number of training examples- and we present a simple incremental approach that allows the practitioner to specify the computational effort to be devoted to this task. We believe that one essential property of probabilistic models, such as the RVM, is the predictive distributions that they provide. Unfortunately, the quality of such distributions depends on that of effective prior over functions. The RVM, with localised basis functions, has a counterintuitive prior over functions, where maximum variation of the function happens at the training inputs. This has the undesirable consequence that the predictive variances are largest at the training inputs, and then shrink as the test input moves away from them. We propose the RVM*, a simple fix to the RVM at prediction time that results in much more appropriate priors over functions, and predictive uncertainties.

Gaussian Processes (GPs) for regression are a general case of RVMs, and a particular case of extended linear models where Gaussian priors are imposed over the weights, and their number grows to infinity. For GPs, the prior is directly specified over function outputs, which are assumed to be jointly Gaussian. The inference task consists in finding the covariance function, which expresses how similar two outputs should be as a function of the similarity between their inputs. GPs are well known for their state of the art performance in regression tasks. Unfortunately, they suffer from high computational costs in training and testing, since these are cubic in the number of training samples for training, and quadratic for predicting. Although one may see RVMs as sparse approximations to GPs, they achieve this in an indirect manner. On the other hand other computationally efficient approaches are explicit approximations to GPs. In Chap. 3 we interest ourselves in one such family of approximations, the Reduced Rank Gaussian Processes (RRGPs). We give an introduction to GPs, and show the fact that in general they are equivalent to extended linear models with infinitely many weights. The RRGP approach is based on approximating the infinite linear model with a finite one, resulting in a model similar in form to the RVM. Learning an RRGP implies solving two tasks: one is the selection of a "support set" (reduced set of inputs) and the second to infer the parameters of the GP covariance function. We address how to solve these tasks, albeit separately, since the joint selection of support set and parameters is a challenging optimisation problem, which poses the risk of over-fitting (fitting the training data too closely, at the expense of a poor generalisation). Like RVMs, RRGPs suffer from poor predictive variances. We propose a modification to RRGPs, similar to that of the RVM*, which greatly improves the predictive distribution.

When presented with pairs of inputs and outputs at training time, or with inputs only at test time, it is very common to consider only the outputs as noisy. This output noise is then explicitly modelled. There situations, however, where it is acceptable to consider the training inputs as deterministic, but it might be essential to take into account the uncertainty in the test inputs. In Chap. 4 we derive the equations for predicting at an uncertain input having Gaussian distribution, with GPs and RVMs. We also present a specific situation that motivated this algorithm: iterated time-series predictions with GPs, where the inputs are composed of previous predictions. Since GPs produce predictive distributions, and those are fed into future inputs to the model, we know that these inputs will be uncertain with known Gaussian distribution. When predicting k-steps ahead, we rely on $k-1$ intermediate predictions, all of which are uncertain. Failing to take into account this accumulated uncertainty implies that the predictive distribution of the k-th prediction is very overconfident. The problem of training a GP when the training inputs are noisy is a harder one, and we address it without the ambition of providing a definitive solution. We propose to approximations to the full integration over uncertain inputs, which is analytically intractable. In a first approach, we maximise the joint posterior over uncertain inputs and GP hyperparameters. This has the interesting consequence of imputing the "true" unseen inputs. However, the optimisation suffers from very many undesirable spurious global maxima, that correspond to extreme over-fitting. For this reason we propose to anneal the output noise, instead of learning it. Since we do not have any satisfactory stopping criterion, previous knowledge of the actual output noise is required, which is unsatisfactory. The second approach consists in sampling from the posterior on the uncertain inputs, while still learning the hyperparameters, in the framework of a "stochastic" EM algorithm. While the results are encouraging, sampling severely increases the already high computational cost of training GPs, which restricts the practical use of the method to rather small training datasets. However, the success on toy examples of this preliminary work does show that there is very exciting work to be pursued in learning with uncertain inputs.

Chapter 2

Sparse Probabilistic Linear Models and the RVM

Linear models form the function outputs by linearly combining a set of inputs (or non-linearly transformed inputs in the general case of "extended" linear models). In the light of some training data, the weights of the model need either to be estimated, or in the Bayesian probabilistic approach a posterior distribution on the weights needs to be inferred.

In traditional data fitting, where the goal is to learn a point estimate of the model weights, it has since long been realised that this estimation process must be accompanied by regularisation. Regularisation consists in forcing the estimated weights to be small in some sense, by adding a penalty term to the objective function which discourages the weights from being large. Regularisation has two beneficial consequences. First, it helps guarantee stable solutions, avoiding the ridiculously large values of the weights that arise from numerical ill-conditioning, and allowing us to solve for the case where there are fewer training examples than weights in the model. Second, by forcing the weights to be smaller than the (finite) training data would suggest, smoother functions are produced which fit the training data somewhat worse, but that fit new unseen test data better. Regularisation therefore helps improve generalisation. The Bayesian framework allows to naturally incorporate the prior knowledge that the weights should be small into the inference process, by specifying a prior distribution. The Bayesian treatment of linear models is well established; O'Hagan (1994, Chap. 9) for example gives an extensive treatment. We introduce probabilistic extended linear models in Sect. 2.1, and establish the connection between Gaussian priors on the model weights and regularisation.

Sparseness has become a very popular concept, mostly since the advent of Support Vector Machines (SVMs), which are sparse extended linear models that excel in classification tasks. A tutorial treatment of SVMs may be found in Schölkopf and Smola (2002). A linear model is sparse if a significant number of its weights is very small or effectively zero. Sparseness offers two key advantages. First, if the number of weights that are non-zero is reduced, the computational cost of making predictions on new test points decreases. Computational cost limits the use of many models in practice. Second, sparseness can be related to regularisation in that models with few non-zero weights produce smoother functions that generalise better. Concerned with sparseness and inspired by the work of MacKay (1994) on prior distributions that automatically select relevant features, Tipping (2001) recently introduced the Relevance Vector Machine (RVM), which is a probabilistic extended linear model with a prior on the weights that enforces sparse solutions. Of course, under a full Bayesian treatment there is no room for sparseness, since there will always be enough posterior probability mass on non-sparse solutions. As with many other models, the full Bayesian RVM is analytically intractable. We present the RVM in Sect. 2.2, and explain how sparseness arises from a specific approximation to the full Bayesian treatment. In Sect. 2.3 we give an example of the use of the RVM for non-linear time series prediction with automatic adaptation of the basis functions. One weak aspect of the RVM is its high training computational cost of $\mathcal{O}(N^3)$, where N is the number of training examples. This has motivated us to propose a very simple incremental approach to training RVMs, the Subspace EM (SSEM) algorithm, which considerably reduces the cost of training, allowing the practitioner to decide how much computational power he wants to devote to this task. We present our SSEM approach in Sect. 2.4.

We believe probabilistic models to be very attractive because they provide full predictive distributions instead of just point predictions. However, in order for these predictive distributions to be sensible, sensible priors over function values need to be specified in the first place, so as to faithfully reflect our beliefs. Too often "convenience" priors are used that fail to fulfil this requirement. In Sect. 2.5 we show that the prior over the weights defined for the RVM implies an inappropriate prior over functions. As a consequence the RVM produces inappropriate predictive uncertainties. To solve this problem, while retaining its nice sparseness properties, we propose a fix at prediction time to the RVM. Our new model, the RVM*, implies more natural priors over functions and produces significantly better predictive uncertainties.

2.1 Extended Linear Models

We will consider extended linear models that map an input Euclidean space of some dimension into a single dimensional Euclidean output space. Given a set of training inputs $\{\mathbf{x}_i | i = 1, \ldots, N\} \subset \mathbb{R}^D$ organised as rows in matrix X, the outputs of an extended linear model are a linear combination of the response of a set of basis functions $\{\phi_j(\mathbf{x}) | j = 1, \ldots, M\} \subset [\mathbb{R}^D \to \mathbb{R}]$:

$$f(\mathbf{x}_i) \; = \; \sum_{j=1}^{M} \phi_j(\mathbf{x}_i)\, w_j = \boldsymbol{\phi}(\mathbf{x}_i)\, \mathbf{w}, \qquad\qquad \mathbf{f} \; = \; \mathbf{\Phi}\, \mathbf{w} \; . \tag{2.1}$$

where $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$ are the function outputs, $\mathbf{w} = [w_1, \ldots, w_M]^\top$ are the weights and $\phi_j(\mathbf{x}_i)$ is the response of the j-th basis function to input x_i. We adopt the following shorthand: $\boldsymbol{\phi}(\mathbf{x}_i) = [\phi_1(\mathbf{x}_i), \ldots, \phi_M(\mathbf{x}_i)]$ is a row vector containing the response of all basis functions to input $\mathbf{x}_i$, $\boldsymbol{\phi}_j = [\phi_j(\mathbf{x}_1), \ldots, \phi_j(x_N)]^\top$ is a column vector containing the response of basis function $\boldsymbol{\phi}_j(\mathbf{x})$ to all training inputs and $\mathbf{\Phi}$ is an $N \times M$ matrix whose j-th column is vector $\boldsymbol{\phi}_j$ and whose i-th row is vector $\phi(\mathbf{x}_i)$. For notational clarity we will not explicitly consider a bias term, i.e. a constant added to the function outputs. This is done without loss of generality, since it would suffice to set one basis function $\boldsymbol{\phi}_{bias}(\mathbf{x}) = 1$ for all $\mathbf{x}$, and the corresponding weight w_{bias} would be the bias term.

The unknowns of the model are the weights $\mathbf{w}$. To estimate their value, one needs a set of training targets $\mathbf{y} = [y_1, \ldots, y_N]^\top$, with each $y_i \subset \mathbb{R}$ associated to its corresponding training input $\mathbf{x}_i$. We will make the common assumption that the observed training outputs differ from the corresponding function outputs by Gaussian iid. noise of variance σ^2:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \qquad\qquad \epsilon_i = \mathcal{N}(0, \sigma^2) \; , \tag{2.2}$$

where it is implicitly assumed that the "true" function can be expressed as an extended linear model. The noise model allows us to write the likelihood of the weights, and its negative logarithm $\mathcal{L}$, which can be used as a target function for estimating $\mathbf{w}$:

$$p(\mathbf{y}|X, \mathbf{w}, \sigma^2) \sim \mathcal{N}(\mathbf{\Phi}\, \mathbf{w}, \sigma^2\, \mathbf{I}), \quad \mathcal{L} = \frac{1}{2}\log(2\pi) + \frac{1}{2}\log\sigma^2 + \frac{1}{2}||\mathbf{y} - \mathbf{\Phi}\, \mathbf{w}||^2, \tag{2.3}$$

where $\mathbf{I}$ is the identity matrix. Maximum Likelihood (ML) estimation of $\mathbf{w}$ can be achieved by minimising $\mathcal{L}$: it is the negative of a monotonic transformation of the likelihood. Taking derivatives and equating to zero one obtains:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = -\mathbf{w}^\top \mathbf{\Phi}^\top \mathbf{y} - \mathbf{w}^\top \mathbf{\Phi}^\top \mathbf{\Phi} = 0 \quad \Rightarrow \quad \mathbf{w} = (\mathbf{\Phi}^\top \mathbf{\Phi})^{-1} \mathbf{\Phi}^\top \mathbf{y}, \tag{2.4}$$

which is the classical expression given by the *normal equations*. This is not surprising: (2.3) shows that maximising the likelihood wrt. $\mathbf{w}$ under a Gaussian noise assumption is equivalent to minimising the sum of squared residuals given by $||\mathbf{y} - \mathbf{\Phi}\,\mathbf{w}||^2$, which is how the normal equations are obtained. A biased estimate of the variance of the noise can be obtained by minimising $\mathcal{L}$. Taking derivatives and equating to zero gives $\sigma^2 = \frac{1}{N}||\mathbf{y} - \mathbf{\Phi}\,\mathbf{w}||^2$: this is the mean of the squared residuals. The unbiased estimate of the variance would divide the sum of squared residuals by $N-1$ instead, corresponding to the number of degrees of freedom.

The normal equations are seldom used as given in (2.4), for at least two reasons. First, notice that if $M > N$, we have an under-complete set of equations and there are infinitely many solutions for $\mathbf{w}$. Matrix $\mathbf{\Phi}^\top\mathbf{\Phi}$ of size $M \times M$ then has at most rank N and can therefore not be inverted. A usual solution is to *regularise* the normal equations, by adding a ridge term controlled by a regularisation parameter λ:

$$\mathbf{w} = (\mathbf{\Phi}^\top\mathbf{\Phi} + \lambda\,\mathbf{I})^{-1}\mathbf{\Phi}^\top\mathbf{y}. \tag{2.5}$$

This is equivalent to minimising a penalised sum of squared residuals $||\mathbf{y} - \mathbf{\Phi}\,\mathbf{w}||^2 + \lambda\,||\mathbf{w}||^2$. Clearly, the regularisation term, $\lambda\,||\mathbf{w}||^2$, penalises large weight vectors and selects from the infinite number of solutions one for which the norm of $\mathbf{w}$ is smallest. The regularised normal equations correspond to Tikhonov regularisation (Tikhonov and Arsenin, 1977) where the smallest eigenvalue of the problem is forced to be λ.

The second reason for regularising the normal equations is to obtain a better generalisation performance, that is to reduce the error made when predicting at new unseen test inputs. An example may help visualise the relation between regularisation and generalisation. Let us consider radial basis functions of squared exponential form:

$$\phi_j(\mathbf{x}_i) = \exp\left(-\frac{1}{2}\sum_{d=1}^{D}(X_{id} - X_{jd})^2/\theta_d^2\right), \tag{2.6}$$

where X_{id} is the d-th component of $\mathbf{x}_i$ and where θ_d is a lengthscale parameter for the d-th dimension. Let us consider a one dimensional input example, and set $\theta_1 = 1$.[1] We generate a training set, shown in Fig. 2.1 by the crosses, by taking 20 equally spaced points in the $[-10, 10]$ interval as inputs. The outputs are generated by applying the 'sinc' function $(\sin(x)/x)$ to the inputs and adding noise of variance 0.01. We decide to use $M = N$ basis functions, centred on the training inputs, and learn the weights of the extended linear model from the normal equations and from the regularised normal equations. Notice that in

[1] We will discuss ways of learning the parameters of the basis functions later in Sect. 2.3.

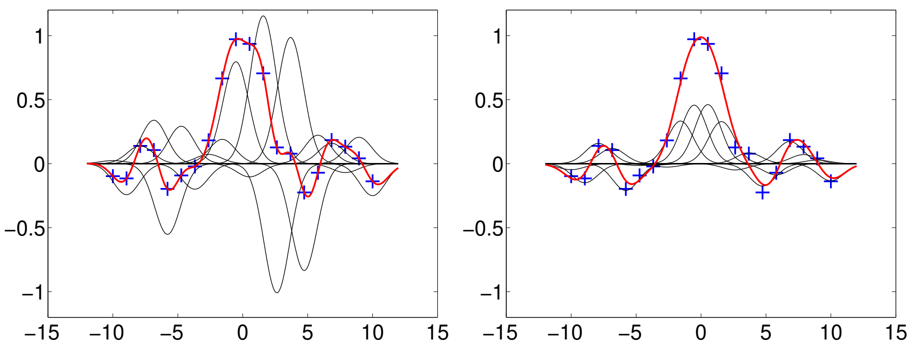

Figure 2.1: Impact of regularisation on generalisation. Non-regularised (*left*) and regularised (*right*) extended linear model with $\lambda = 1$. The training data is represented by the crosses, and the thin lines are the basis functions multiplied by their corresponding weights. The thick lines are the functions given by the extended linear model, obtained by adding up the weighted basis functions.

this example regularisation is not needed for numerical reasons: $\mathbf{\Phi}^\top \mathbf{\Phi}$ can safely be inverted. In the left pane of Fig. 2.1 We present the basis functions weighted by the $\mathbf{w}$ obtained from the non-regularised normal equations (thin lines), and the corresponding function (thick line) obtained by adding them up. The right pane represents the same quantities for the regularised case. The weights in the regularised case are smaller, and the response of the model is smoother and seems to over-fit less than in the non-regularised case. The mean square error on a test set with 1000 inputs equally spaced in the $[-12, 12]$ interval is 0.066 without regularisation versus 0.033 with regularisation. Regularisation forces the weights to be small, giving smoother models that generalise better.

Two questions arise: first, we know how to estimate $\mathbf{w}$ and σ^2, but only if the regularisation parameter λ is given. How can λ be learned? Certainly not by minimising the penalised sum of squared residuals $||\mathbf{y} - \mathbf{\Phi}\,\mathbf{w}||^2 + \lambda\,||\mathbf{w}||^2$, since this would give a trivial solution of $\lambda = 0$. We address this question in the next section, where we show that a simple Bayesian approach to the extended linear model gives rise to a regularisation term as in (2.5). The second question is why the penalisation term, $\lambda\,||\mathbf{w}||^2$, is in terms of the squared 2-norm of $\mathbf{w}$. One reason is analytic convenience: it allows to obtain the regularised normal equations. Other penalisation terms have been proposed, the 1-norm case being a very popular alternative (Tibshirani, 1996). While the 2-norm penalty term uniformly reduces the magnitude of all the weights, the 1-norm has the property of shrinking a selection of the weights and of therefore giving sparse solutions. Sparseness will be a central issue of this chapter, and we will see in Sect. 2.2

that it can also arise when using 2-norm penalty terms in a Bayesian setting with hierarchical priors.

2.1.1 A Bayesian Treatment

The Bayesian approach to learning provides with an elegant framework where prior knowledge (or uncertainty) can directly be expressed in terms of probability distributions, and incorporated into the model. Let us consider what we have learned from the previous section: for an extended linear model, forcing the weights to be small gives smoother functions with better generalisation performance.

This prior knowledge about the weights can be expressed by treating them as random variables and defining a prior distribution that expresses our belief about $\mathbf{w}$ before we see any data. One way of expressing our knowledge is to impose that every weight be independently drawn from the same Gaussian distribution, with zero mean and variance ${\sigma^2}_w$:

$$p(w_j|\sigma_w^2) \sim \mathcal{N}(0, \sigma_w^2) \ , \qquad p(\mathbf{w}|\sigma_w^2) \sim \mathcal{N}(0, \sigma_w^2 \, \mathbf{I}) \ . \tag{2.7}$$

Our knowledge about the weights will be modified once we observe data. The data modifies the prior through the likelihood of the observed targets, and leads to the posterior distribution of the weights via Bayes rule:

$$p(\mathbf{w}|\mathbf{y}, X, \sigma_w^2, \sigma^2) = \frac{1}{Z} \, p(\mathbf{y}|X, \mathbf{w}, \sigma^2) \, p(\mathbf{w}|\sigma_w^2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ , \tag{2.8}$$

where $Z = p(\mathbf{y}|X, \sigma_w^2, \sigma^2)$ is here a normalisation constant about which we will care later, when we consider learning σ_w^2 and σ^2. Since both the likelihood and the prior are Gaussian in $\mathbf{w}$, the posterior is also a Gaussian distribution with covariance $\boldsymbol{\Sigma}$ and mean $\boldsymbol{\mu}$, respectively given by:

$$\boldsymbol{\Sigma} = (\sigma^{-2} \, \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \sigma_w^{-2} \, \mathbf{I})^{-1} \ , \qquad \boldsymbol{\mu} = \sigma^{-2} \boldsymbol{\Sigma} \, \boldsymbol{\Phi}^\top \mathbf{y} \ . \tag{2.9}$$

In the previous section, given some data we learned one value of the weights for a particular regularisation constant. Now, once we observe data and for a given noise and prior variance of the weights, we obtain a posterior distribution on $\mathbf{w}$. The extended linear model has thus become probabilistic in that function values are now a linear combination of random variables $\mathbf{w}$. For a given a new test input $\mathbf{x}_*$, instead of a point prediction we now obtain a predictive distribution $p(f(\mathbf{x}_*)|\mathbf{x}_*, \mathbf{y}, X, \sigma_w^2, \sigma^2)$, which is Gaussian[2] with mean and variance given by:

$$m(\mathbf{x}_*) = \boldsymbol{\phi}(\mathbf{x}_*) \, \boldsymbol{\mu} \ , \qquad v(\mathbf{x}_*) = \boldsymbol{\phi}(\mathbf{x}_*) \, \boldsymbol{\Sigma} \, \boldsymbol{\phi}(\mathbf{x}_*)^\top \ . \tag{2.10}$$

[2] $f(\mathbf{x}_*)$ is a linear combination of the weights, which have a Gaussian posterior distribution.

We will later analyse the expression for the predictive variance, and devote to it the whole of Sect. 2.5. The predictive mean is often used as a point prediction, which is optimal for quadratic loss functions (in agreement with the Gaussian noise assumption we have made). We observe that the predictive mean is obtained by estimating the weights by the maximum of their posterior distribution. This is also called the Maximum A Posteriori (MAP) estimate of the weights. Since the model is linear in the weights and the posterior is a symmetric unimodal distribution, it is clear that this would be the case. The MAP estimate of the weights corresponds to the mean of their posterior, which can be rewritten as:

$$\boldsymbol{\mu} = \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \frac{\sigma^2}{\sigma_w^2} \mathbf{I} \right)^{-1} \boldsymbol{\Phi}^\top \mathbf{y} \ . \tag{2.11}$$

This expression is identical to that of the regularised normal equations, (2.5), if we set the regularisation parameter to $\lambda = \sigma^2/\sigma_w^2$. The regularisation parameter is inversely proportional to the signal-to-noise ratio (SNR), since the variance of the extended linear model is proportional to the prior variance of the weights σ_w^2. The larger the noise relative to the prior variance of the function, the smaller the weights are forced to be, and the smoother the model.

Under a correct Bayesian approach, using the results described thus far implies prior knowledge of σ^2 and of σ_w^2. However, it is reasonable to assume that we do not possess exact knowledge of these quantities. In such case, we should describe any knowledge we have about them in terms of a prior distribution, obtain a posterior distribution and integrate them out at prediction time. The disadvantage is that in the vast majority of the cases the predictive distribution will not be Gaussian anymore, and will probably not even be analytically tractable. Consider the noisy version y_* of $f(\mathbf{x}_*)$. With knowledge of σ_w^2 and σ^2, the predictive distribution of $\mathbf{y}_*$ is $p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \sigma_w^2, \sigma^2) \sim \mathcal{N}(m(\mathbf{x}_*), v(\mathbf{x}_*) + \sigma^2)$. Now, given a posterior distribution $p(\sigma_w^2, \sigma^2|\mathbf{y}, X)$ this predictive distribution becomes:

$$p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X) = \int p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \sigma_w^2, \sigma^2)\, p(\sigma_w^2, \sigma^2|\mathbf{y}, X) \mathrm{d}\, \sigma_w^2 \, \mathrm{d}\, \sigma^2 \ . \tag{2.12}$$

Approximating the posterior on σ_w^2 and σ^2 by a delta function at its mode, $p(\sigma_w^2, \sigma^2|\mathbf{y}, X) \approx \delta(\hat{\sigma}_w^2, \hat{\sigma}^2)$ where $\hat{\sigma}_w^2$ and $\hat{\sigma}^2$ are the maximisers of the posterior, brings us back to the simple expressions of the case where we had knowledge of σ_w^2 and σ^2. The predictive distribution is approximated by replacing the values for σ_w^2 and σ^2 by their MAP estimates: $p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X) \approx p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \hat{\sigma}_w^2, \hat{\sigma}^2)$. The goodness of this approximation is discussed for example by Tipping (2001).

Under this approximate Bayesian scheme we need to learn the MAP values of σ_w^2 and σ^2, which is equivalent to maximising their posterior, obtained again by applying Bayes rule:

$$p(\sigma_w^2, \sigma^2 | \mathbf{y}, X) \propto p(\mathbf{y}|X, \sigma_w^2, \sigma^2)\, p(\sigma_w^2, \sigma^2) \ . \tag{2.13}$$

If one now decides to use an uninformative improper uniform prior $p(\sigma_w^2, \sigma^2)$, maximising the posterior becomes equivalent to maximising the marginal likelihood $p(\mathbf{y}|X, \sigma_w^2, \sigma^2)$ which was the normalising constant in (2.8). The marginal likelihood, also called *evidence* by MacKay (1992),

$$p(\mathbf{y}|X, \sigma_w^2, \sigma^2) = \int p(\mathbf{y}|X, \mathbf{w}, \sigma^2)\, p(\mathbf{w}|\sigma_w^2)\, \mathrm{d}\mathbf{w} \sim \mathcal{N}(0, \mathbf{K}) \ , \tag{2.14}$$

has the nice property of being Gaussian since it is the integral of the product of two Gaussians in $\mathbf{w}$. It has zero mean since $\mathbf{y}$ is linear in $\mathbf{w}$, and the prior on $\mathbf{w}$ has zero mean. Its covariance is given by $\mathbf{K} = \sigma^2\, \mathbf{I} + \sigma_w^2\, \Phi\, \Phi^\top$. Maximisation of the marginal likelihood to infer the value of σ_w^2 and σ^2 is also referred to as Maximum Likelihood II (MLII). Unfortunately, there is no closed form solution to this maximisation in our case. Equivalently, it is the logarithm of the marginal likelihood that is maximised:

$$\mathcal{L} = \log p(\mathbf{y}|X, \sigma_w^2, \sigma^2) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}| - \frac{1}{2\sigma^2}\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y} \ , \tag{2.15}$$

with derivatives:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \sigma_w^2} &= -\frac{1}{2}\,\mathrm{Tr}(\Phi\, \Phi^\top\, \mathbf{K}^{-1}) + \frac{1}{2\sigma^2}\,\mathbf{y}^\top \mathbf{K}^{-1} \Phi\, \Phi^\top \mathbf{K}^{-1}\mathbf{y} \ , \\ \frac{\partial \mathcal{L}}{\partial \sigma^2} &= -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\,\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y} \ . \end{aligned} \tag{2.16}$$

It is common to use a gradient descent algorithm to minimise the negative of $\mathcal{L}$. For the example of Fig. 2.1, minimising $\mathcal{L}$ using conjugate gradients we obtain $\hat{\sigma}_w^2 = 0.0633$ and $\hat{\sigma}^2 = 0.0067$. This translates into a regularisation constant, $\lambda = 0.1057$, which coincidentally is fairly close to our initial guess.

2.2 The Relevance Vector Machine

The Relevance Vector Machine (RVM), introduced by Tipping (2001) (see also Tipping, 2000) is a probabilistic extended linear model of the form given by (2.1), where as in Sect. 2.1.1 a Bayesian perspective is adopted and a prior distribution is defined over the weights. This time an individual Gaussian prior is defined over each weight independently:

$$p(w_j|\alpha_j) \sim \mathcal{N}(0, \alpha_j) \ , \qquad p(\mathbf{w}|\mathbf{A}) \sim \mathcal{N}(0, \mathbf{A}) \ , \tag{2.17}$$

which results in an independent joint prior over $\mathbf{w}$, with separate precision hyperparameters $\mathbf{A} = \operatorname{diag}\boldsymbol{\alpha}$, with $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]$. [3] For given $\boldsymbol{\alpha}$, the RVM is then identical to our extended linear model of the previous section, where $\sigma_w^2\,\mathbf{I}$ is now replaced by $\mathbf{A}$.

Given $\boldsymbol{\alpha}$ and σ^2, the expressions for the mean and variance of the predictive distribution of the RVM at a new test point $\mathbf{x}_*$ are given by (2.10). The Gaussian posterior distribution over $\mathbf{w}$ is computed as in previous section, but its covariance is now given by:

$$\boldsymbol{\Sigma} = (\sigma^{-2}\,\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \mathbf{A})^{-1} \ , \tag{2.18}$$

and its mean (and MAP estimate of the weights) by:

$$\boldsymbol{\mu} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma^2\,\mathbf{A}\right)^{-1} \boldsymbol{\Phi}^\top\mathbf{y} \ , \tag{2.19}$$

which is the minimiser of a new form of penalised sum of squared residuals $||\mathbf{y} - \boldsymbol{\Phi}\,\mathbf{w}||^2 + \sigma^2\,\mathbf{w}^\top\mathbf{A}\,\mathbf{w}$. Each weight is individually penalised by its associated hyperparameter α_j: the larger the latter, the smaller the weight w_j is forced to be. Sparse models can thus be obtained by setting some of the α's to very large values (or effectively infinity). This has the effect setting the corresponding weights to zero, and therfore of "pruning" the corresponding basis functions from the extended linear model. The basis functions that remain are called the Relevance Vectors (RVs). Sparsity is a key characteristic of the RVM, and we will later see how it arises, due to the particular way in which α is learned. Like in previous section, in the RVM framework the posterior distribution of $\boldsymbol{\alpha}$ and σ^2 is approximated by a delta function centred on the MAP estimates. This predictive distribution of the RVM is therefore derived as we have just done, by assuming $\boldsymbol{\alpha}$ and σ^2 known, given by their MAP estimates.

Tipping (2001) proposes the use of Gamma priors on the α's and of an inverse Gamma prior on σ^2. In practice, he proposes to take the limit case of improper uninformative uniform priors. A subtle point needs to be made at this point. In practice, since $\boldsymbol{\alpha}$ and σ^2 need to be positive, it is the posterior distribution of their logarithms that is considered and maximised, given by $p(\log\boldsymbol{\alpha}, \log\sigma^2|\mathbf{y}, X) \propto p(\mathbf{y}|X, \log\boldsymbol{\alpha}, \log\sigma^2)\,p(\log\alpha, \log\sigma^2)$. For MAP estimation to be equivalent to MLII, uniform priors are defined over a *logarithmic* scale. This of course implies non-uniform priors over a linear scale, and is ultimately responsible for obtaining sparse solutions, as we will discuss in Sect. 2.2.2. Making inference for the RVM corresponds thus to learning $\boldsymbol{\alpha}$ and σ^2 by MLII, that is by maximising the marginal likelihood in the same manner as we described in the previous section. The RVM marginal likelihood (or evidence) is

[3] We have decided to follow Tipping (2001) in defining prior precisions instead of prior variances, as well as to follow his notation.

given by:

$$p(\mathbf{y}|X, \boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{y}|X, \mathbf{w}, \sigma^2)\, p(\mathbf{w}|\boldsymbol{\alpha})\, \mathrm{d}\mathbf{w} \sim \mathcal{N}(0, \mathbf{K}) \ , \tag{2.20}$$

where the covariance is now $\mathbf{K} = \sigma^2\, \mathbf{I} + \boldsymbol{\Phi}\, \mathbf{A}^{-1}\, \boldsymbol{\Phi}^\top$. It is the logarithm of the evidence that is in practice maximised (we will equivalently minimise its negative), given by (2.15) with the RVM covariance.[4] In practice, one can exploit that $M < N$ because of sparseness to compute the negative log evidence in a computationally more efficient manner (see Tipping, 2001, App. A):

$$\mathcal{L} = \frac{1}{2}\log(2\pi) + \frac{N}{2}\log\sigma^2 - \frac{1}{2}\log|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{j=1}^{M}\log\alpha_j + \frac{1}{2\sigma^2}\mathbf{y}^\top(\mathbf{y} - \boldsymbol{\Phi}\,\boldsymbol{\mu}) \ . \tag{2.21}$$

One approach to minimising the negative log evidence is to compute derivatives, this time with respect to the log hyperparameters:

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\log\alpha_j} &= -\frac{1}{2} + \frac{1}{2}\alpha_j\,(\boldsymbol{\mu}_j^2 + \boldsymbol{\Sigma}_{jj}) \ , \\ \frac{\partial\mathcal{L}}{\partial\log\sigma^2} &= \frac{N}{2} - \frac{1}{2\sigma^2}\left(\mathrm{Tr}[\boldsymbol{\Sigma}\,\boldsymbol{\Phi}^\top\boldsymbol{\Phi}] + ||\mathbf{y} - \boldsymbol{\Phi}\,\boldsymbol{\mu}||^2\right) \ , \end{aligned} \tag{2.22}$$

and to again use a gradient descent algorithm. At this point one may wonder whether the negative log evidence, (2.21), is convex. It has been shown by Faul and Tipping (2002) that for a fixed estimate of the noise variance, the Hessian of the log evidence wrt. $\boldsymbol{\alpha}$ is positive semi-definite (the fact that it is not strictly positive has to do with infinity being a stationary point for some of the α's). Therefore, given the noise, $\mathcal{L}$ is only convex as a function of a single α given the rest, and has multiple minima as a function of $\boldsymbol{\alpha}$. As a function both of $\boldsymbol{\alpha}$ and σ^2 the negative log evidence has also multiple minima, as we show in a simple example depicted in Fig. 2.2. We use the same toy data as in the example in Fig. 2.1 (one dimensional input space, noisy training outputs generated with a 'sinc' function) and the same form of squared exponential basis functions centred on the 20 training inputs. We minimise $\mathcal{L}$ 100 times wrt. $\boldsymbol{\alpha}$ and σ^2 from different random starting points using conjugate gradients. We present the result of each minimisation in Fig. 2.2 by a patch. The position in along the x-axis shows the number of RVs, and that along the y-axis the value of $\mathcal{L}$ at the minimum attained. The radius of the patch is proportional to the test mean squared error of the model obtained, and the colour indicates the logarithm base 10 of the estimated σ^2. The white patches with black contour and a solid circle inside are solutions for which the estimated σ^2 was much smaller than for the other solutions (more than 5 orders of magnitude smaller); it would have been

[4] Interestingly, all models considered in this Thesis have log marginal likelihood given by (2.15): the differentiating factor is the form of the covariance matrix $\mathbf{K}$.

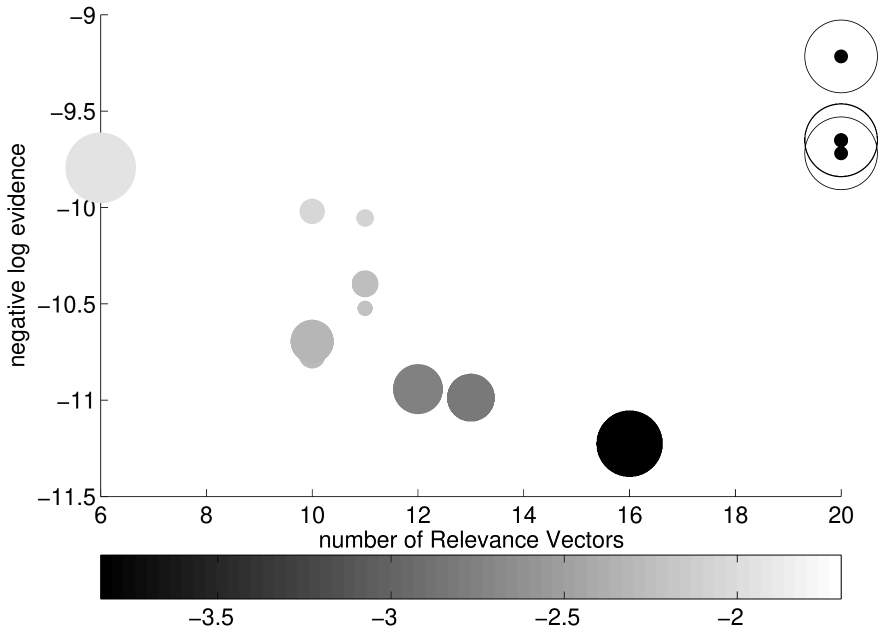

Figure 2.2: Multiple local minima of the RVM negative log evidence. Every patch represents one local minimum. The *radius* is proportional to the test mean squared error of the corresponding model. The *colour* indicates the magnitude of the estimated noise variance σ^2 in a logarithm base 10 scale. The white patches with black contour and a solid black circle inside correspond to cases where the estimated σ^2 was very small ($< 10^{-9}$) and it was inconvenient to represent it with our colourmap.

inconvenient to represent them in the same colour map as the other solutions, but you should think of them as being really dark. The reason why less than 100 patches can be observed is that identical local minima are repeatedly reached. Also, there is no reason to believe that other local minima do not exist, we may just never have reached them.

We observe that the smaller the estimate of the noise, the more RVs are retained. This makes sense, since small estimates of σ^2 imply that high model complexity is needed to fit the data closely. Sparse solutions are smooth, which fits well with a higher noise estimate. Except for the extreme cases with very small σ^2, the more RVs and the smaller noise, the smaller the negative log evidence. The extreme cases of tiny noise correspond to poor minima; the noise is too small compared to how well the data can be modelled, even with full model complexity. These, it seems to us, are undesirable minima, where probably the $\log|\sigma^2\,\mathbf{I}+\mathbf{\Phi}\,\mathbf{A}^{-1}\,\mathbf{\Phi}^\top|$ term of $\mathcal{L}$ has enough strength locally to keep the noise from growing to reasonable values. The smallest minimum of $\mathcal{L}$ was found for 16 RVs and $\sigma^2 \approx 10^{-4}$. The corresponding model, however, has a quite poor test error. The model with smallest mean squared test error (0.029, versus 0.033 for the regularised normal equations in previous section) is obtained for 11 RVs, with $\mathcal{L} \approx -10.5$, above the smallest minimum visited. We plot both solutions in Fig. 2.3. Indeed, the experiment shows that a small negative log evidence does not imply good generalisation, and that learning by minimising it leads to over-fitting: the training data are too well approximated, at the expense of poor performance on new unseen test points. We believe over-fitting is a direct consequence of MLII learning with an overly complex hyperprior. Sparseness may save the day, but it only seems to arise when the estimated noise is reasonably large. If MLII is the method to be used, it could be sensible not to use a uniform prior (over a logarithmic scale) for σ^2, but rather one that clearly discourages too small noises. Priors that enforce sparseness are not well accepted by Bayesian purists, since pruning is in disagreement with Bayesian theory. See for example the comments of Jaynes (2003, p. 714) on the work of O'Hagan (1977) on outliers. Sparseness is nevertheless popular these days, as the popularity of Support Vector Machines (SVMs) ratifies (see for example the recent book by Schölkopf and Smola, 2002), and constitutes a central characteristic of the RVM, which we can now see as a semi-Bayesian extended linear model. We devote Sect. 2.2.2 to understanding why sparsity arises in the RVM.

2.2.1 EM Learning for the RVM

Tipping (2001) does not suggest direct minimisation of the negative log evidence for training the RVM, but rather the use of an approximate Expectation-Maximisation (EM) procedure (Dempster et al., 1977). The integration with respect to the weights to obtain the marginal likelihood makes the use of the EM algorithm with the weights as "hidden" variables very natural. Since it is most common to use the EM for maximisation, $\mathcal{L}$ will now become the (positive) log evidence; we hope that the benefit of convenience will be superior to the confusion cost of this change of sign. To derive the EM algorithm for the RVM we first obtain the log marginal likelihood by marginalising the weights from the joint distribution of $\mathbf{y}$ and $\mathbf{w}$:

$$\mathcal{L} = \log p(\mathbf{y}|X,\boldsymbol{\alpha},\sigma^2) = \log \int p(\mathbf{y},\mathbf{w}|X,\boldsymbol{\alpha},\sigma^2)\,\mathrm{d}\mathbf{w} \ , \tag{2.23}$$

where the joint distribution over $\mathbf{y}$ and $\mathbf{w}$ is both equal to the likelihood times

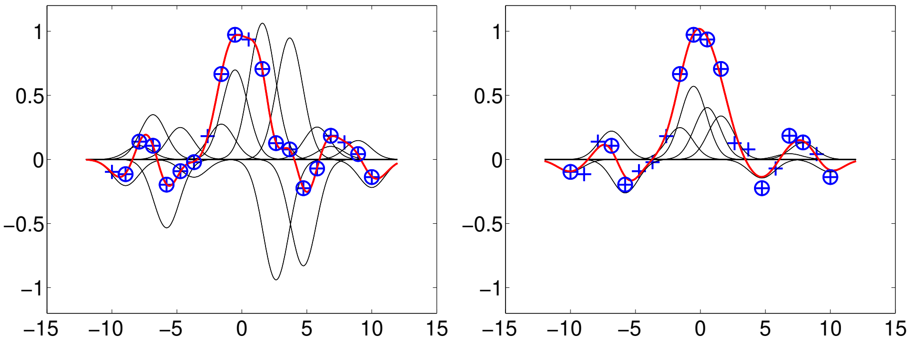

Figure 2.3: RVM models corresponding to two minima of the negative log likelihood. Impact of regularisation on generalisation. *Left*: global minimum found, $\mathcal{L} \approx -11.2$, $\sigma^2 \approx 10^{-4}$ and 16 RVs, corresponds to over-fitting. (*Right*): model with best generalisation performance, $\mathcal{L} \approx -10.5$, $\sigma^2 \approx 10^{-2}$ and 11 RVs. The training data is represented by the crosses, and the thin lines are the basis functions multiplied by their corresponding weights. The thick lines are the functions given by the extended linear model, obtained by adding up the weighted basis functions. The training points corresponding to the RVs are shown with a circle on top.

the prior on $\mathbf{w}$ and to the marginal likelihood times the posterior on $\mathbf{w}$:

$$p(\mathbf{y}, \mathbf{w}|X, \boldsymbol{\alpha}, \sigma^2) = p(\mathbf{y}|X, \mathbf{w}, \boldsymbol{\alpha}, \sigma^2)\, p(\mathbf{w}|\boldsymbol{\alpha}) = p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\alpha}, \sigma^2)\, p(\mathbf{y}|X, \boldsymbol{\alpha}, \sigma^2) \ . \tag{2.24}$$

By defining a "variational" probability distribution $q(\mathbf{w})$ over the weights and using Jensen's inequality, a lower bound on $\mathcal{L}$ can be obtained:

$$\begin{aligned} \mathcal{L} &= \log \int q(\mathbf{w}) \, \frac{p(\mathbf{y}, \mathbf{w}|\boldsymbol{\alpha}, \sigma^2)}{q(\mathbf{w})} \mathrm{d}\mathbf{w} \ , \\ &\geq \int q(\mathbf{w}) \log \frac{p(\mathbf{y}, \mathbf{w}|\boldsymbol{\alpha}, \sigma^2)}{q(\mathbf{w})} \, \mathrm{d}\mathbf{w} \equiv \mathcal{F}(q, \sigma^2, \boldsymbol{\alpha}) \ , \end{aligned} \tag{2.25}$$

The EM algorithm consists in iteratively maximising the lower bound $\mathcal{F}(q, \sigma^2, \boldsymbol{\alpha})$ (or $\mathcal{F}$ for short), and can be seen as a coordinate ascent technique. In the Expectation step (or E-step) $\mathcal{F}$ is maximised wrt. the variational distribution $q(\mathbf{w})$ for fixed parameters $\boldsymbol{\alpha}$ and σ^2, and in the Maximisation step (M-step) $\mathcal{F}$ is maximised wrt. to the hyperparameters $\boldsymbol{\alpha}$ and σ^2 for fixed $q(\mathbf{w})$.

Insight into how to perform the E-step can be gained by re-writing the lower bound $\mathcal{F}$ as:

$$\mathcal{F} = \mathcal{L} + \mathcal{D}\left(q(\mathbf{w})||p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\alpha}, \sigma^2)\right) \quad , \tag{2.26}$$

where $\mathcal{D}\left(q(\mathbf{w})||p(\mathbf{w}|\mathbf{t}, X, \boldsymbol{\alpha}, \sigma^2)\right)$ is the Kullback-Leibler (KL) divergence between the variational distribution $q(\mathbf{w})$ and the posterior distribution of the weights. The KL divergence is always a positive number and is only equal to zero if the two distributions it takes as arguments are identical. The E-step corresponds thus to making $q(\mathbf{w})$ equal to the posterior on $\mathbf{w}$, which implies $\mathcal{F} = \mathcal{L}$,[5] or an "exact" E-step. Since the posterior is Gaussian, the E-step reduces to computing its mean and covariance, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, given by (2.19) and (2.18).

To perform the M-step, it is useful to rewrite $\mathcal{F}$ in a different manner:

$$\mathcal{F} = \int q(\mathbf{w}) \, \log p(\mathbf{y}, \mathbf{w}|X, \boldsymbol{\alpha}, \sigma^2) \, \mathrm{d}\mathbf{w} - \mathcal{H}\left(q(\mathbf{w})\right) \quad , \tag{2.27}$$

where $\mathcal{H}\left(q(\mathbf{w})\right)$ is Shannon's entropy of $q(\mathbf{w})$, which is an irrelevant constant since it does not depend on $\boldsymbol{\alpha}$ or σ^2. The M-step consists therefore in maximising the average of the log joint distribution of $\mathbf{y}$ and $\mathbf{w}$ over $q(\mathbf{w})$ with respect to the parameters σ^2 and $\boldsymbol{\alpha}$; this quantity is given by:

$$\begin{aligned}\int q(\mathbf{w}) \, \log p(\mathbf{y}, \mathbf{w}|X, \boldsymbol{\alpha}, \sigma^2) \, \mathrm{d}\mathbf{w} = \frac{1}{2} \log |\mathbf{A}| - \frac{1}{2} \operatorname{Tr}\left[\mathbf{A}\,\boldsymbol{\Sigma} + \mathbf{A}\boldsymbol{\mu}\,\boldsymbol{\mu}^\top\right] \\ - \frac{N}{2} \log \sigma^2 - \frac{1}{2\sigma^2}\left(\|\mathbf{y} - \boldsymbol{\Phi}\,\boldsymbol{\mu}\|^2 + \operatorname{Tr}[\boldsymbol{\Sigma}\,\boldsymbol{\Phi}^\top\boldsymbol{\Phi}]\right) \quad ,\end{aligned} \tag{2.28}$$

and computing its derivatives wrt. $\boldsymbol{\alpha}$ and σ^2 is particularly simple, since the "trick" of the EM algorithm is that now $q(\mathbf{w})$ and therefore $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are fixed and do not depend on $\boldsymbol{\alpha}$ or σ^2. We get:

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \alpha_j} &= \frac{1}{2\alpha_j} - \frac{1}{2}\left(\boldsymbol{\Sigma}_{jj} + \boldsymbol{\mu}_j^2\right) \quad , \\ \frac{\partial \mathcal{F}}{\partial \sigma^2} &= -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}\left(\operatorname{Tr}[\boldsymbol{\Sigma}\,\boldsymbol{\Phi}^T\boldsymbol{\Phi}] + \|\mathbf{y} - \boldsymbol{\Phi}\,\boldsymbol{\mu}\|^2\right) \quad .\end{aligned} \tag{2.29}$$

Observe that if we want to take derivatives wrt. to the logarithm of $\boldsymbol{\alpha}$ and σ^2 instead, all we need to do is multiply the derivatives we just obtained by α_j and σ^2 respectively. The result are the derivatives of the log marginal likelihood obtained in the previous section (taking into account the sign change of $\mathcal{L}$ from there to here), given by (2.22). Surprising? Not if we look back at (2.26) and note that after the exact E-step the KL divergence between $q(\mathbf{w})$ and the

[5] In some cases the posterior is a very complicated expression, and simpler variational distributions are chosen which do not allow for $\mathcal{F} = \mathcal{L}$. In our case, the posterior is Gaussian, allowing us to match it by choosing a Gaussian $q(\mathbf{w})$.

posterior on $\mathbf{w}$ is zero, and $\mathcal{F} = \mathcal{L}$. The new fact is that here $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ being fixed, it is possible to equate the derivatives to zero and obtain a closed form solution. This gives the update rules:

$$\alpha_j^{\text{new}} = \frac{1}{\boldsymbol{\mu}_j^2 + \boldsymbol{\Sigma}_{jj}} \;, \qquad (\sigma^2)^{\text{new}} = \frac{\|\mathbf{t} - \boldsymbol{\Phi}\,\boldsymbol{\mu}\|^2 + \mathrm{Tr}[\boldsymbol{\Phi}^T \boldsymbol{\Phi}\,\boldsymbol{\Sigma}]}{N} \;, \tag{2.30}$$

which are the same irrespective of whether derivatives were taken in a logarithmic scale or not. The update rule for the noise variance can be expressed in a different way:

$$(\sigma^2)^{\text{new}} = \frac{\|\mathbf{y} - \boldsymbol{\Phi}\,\boldsymbol{\mu}\|^2 + (\sigma^2)^{\text{old}} \sum_j \gamma_j}{N} \;, \tag{2.31}$$

by introducing the quantities $\gamma_j \equiv 1 - \alpha_j \boldsymbol{\Sigma}_{jj}$, which are a measure of how "well-determined" each weight ω_j is by the data (MacKay, 1992).

The EM algorithm for the RVM is guaranteed to increase $\mathcal{L}$ at each step, since the E-step sets $\mathcal{F} = \mathcal{L}$, and the M-step increases $\mathcal{F}$. Tipping (2001, App. A) proposes an alternative update rule for the M-step, that does not locally maximise $\mathcal{F}$. However, this update rule gives faster convergence than the optimal one.[6] The modified M-step, derived by Tipping (2001), is obtained by an alternative choice of independent terms in the derivatives, as is done by MacKay (1992):

$$\alpha_j^{\text{new}} = \frac{\gamma_j}{\boldsymbol{\mu}_j^2} \;, \qquad (\sigma^2)^{\text{new}} = \frac{\|\mathbf{y} - \boldsymbol{\Phi}\,\boldsymbol{\mu}\|^2}{N - \sum_j \gamma_j} \;. \tag{2.32}$$

2.2.2 Why Do Sparse Solutions Arise?

Faul and Tipping (2002) and Wipf et al. (2004) have recently proposed two quite different ways of understanding why the RVM formulation leads to sparse solutions. The first study the location of the maxima of the log marginal likelihood, while the second show that the α's are variational parameters of an approximation to a heavy tailed prior on $\mathbf{w}$; this variational approximation favours sparse solutions.

Faul and Tipping (2002) show that as a function of a single α_i, given the other α's, the marginal likelihood has a unique maximum which can be computed in closed-form. The optimal value of α_i can either be $+\infty$ or a finite value. They also show that the point where all individual log evidences conditioned on one single α_i are maximised is a joint maximum over $\boldsymbol{\alpha}$, since the Hessian

[6] Coordinate ascent algorithms, such as the EM, are sometimes very slow, and incomplete steps make them sometimes faster.

of $\mathcal{L}$ is negative semi-definite. Sparse solutions arise then from the fact that $\mathcal{L}$ has local maxima where some of the α's are genuinely infinity. There is an unsatisfactory ambiguity, however, at this point. The reason why learning $\boldsymbol{\alpha}$ is based on maximising the log evidence (MLII learning) stems from the fact that we are finding the MAP estimate of $\boldsymbol{\alpha}$ under a uniform prior over $\boldsymbol{\alpha}$. Yet MAP solutions are variant under a change of parameterisation. To see this, let us consider the noise known and write the posterior on $\boldsymbol{\alpha}$:

$$p(\boldsymbol{\alpha}|\mathbf{y}, X) \propto p(\mathbf{y}|X, \boldsymbol{\alpha})\, p_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \ , \tag{2.33}$$

where $p_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$ is the prior on $\boldsymbol{\alpha}$. Consider now a change of variable by means of an invertible function $g(\cdot)$. The posterior on $g(\boldsymbol{\alpha})$ is given by:[7]

$$\begin{aligned} p\left(g(\boldsymbol{\alpha})|\mathbf{y}, X\right) &\propto p\left(\mathbf{y}|X, g(\boldsymbol{\alpha})\right)\, p_{g(\boldsymbol{\alpha})}\left(g(\boldsymbol{\alpha})\right) \ , \\ &\propto p\left(\mathbf{y}|X, g(\boldsymbol{\alpha})\right)\, \frac{1}{g'(\boldsymbol{\alpha})}\, p_{\boldsymbol{\alpha}}(\boldsymbol{\alpha}) \ , \end{aligned} \tag{2.34}$$

and we can see that in general $[g(\boldsymbol{\alpha})]_{\mathrm{MAP}} \neq g(\boldsymbol{\alpha}_{\mathrm{MAP}})$.[8] More specifically, for flat priors, the MAP solution depends on the scale on which the prior is defined. For example, we have previously noted that it is convenient to maximise the prior over $\log \boldsymbol{\alpha}$,[9] since this allows to use unconstrained optimisation. To still have that MAP is equivalent to maximum evidence we need to redefine the prior on $\boldsymbol{\alpha}$, so that it is uniform in the logarithmic scale. To exemplify this we give in Table 2.1 the expression of the log posterior on $\boldsymbol{\alpha}$ and on $\log \boldsymbol{\alpha}$ for priors flat in both scales. Maximising the posterior is equivalent to maximising the evidence only if the prior is flat in the scale in which the maximisation is performed. Flat priors are therefore not uninformative for MAP estimation. For the RVM, sparse solutions arise only when maximising the posterior over $\log \boldsymbol{\alpha}$, with a prior flat in $\log \boldsymbol{\alpha}$ (which corresponds in the linear scale to an improper prior that concentrates an infinite amount of mass on sparse solutions).

Wipf et al. (2004) propose a different interpretation of the sparsity mechanism, that has the advantage of not suffering from the MAP ambiguity. The marginal (or "true", indistinctly) prior on $\mathbf{w}$ is considered, which is obtained by marginalising the Gaussian conditional prior $p(\mathbf{w}|\boldsymbol{\alpha})$ over $\boldsymbol{\alpha}$:

$$p_{\mathbf{w}}(\mathbf{w}) = \int p(\mathbf{w}|\boldsymbol{\alpha})\, p(\boldsymbol{\alpha})\, \mathrm{d}\boldsymbol{\alpha} \ , \tag{2.35}$$

which is invariant to changes of representation of $\boldsymbol{\alpha}$. Of course the motivation for using the conditional prior $p(\mathbf{w}|\boldsymbol{\alpha})$ in the RVM setting was that it allowed

[7] Given $p_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$, the prior on $g(\boldsymbol{\alpha})$ is given by $p_{g(\boldsymbol{\alpha})}\left(g(\boldsymbol{\alpha})\right) = p_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})/g'(\boldsymbol{\alpha})$.

[8] This is not surprising, since in general the maximum of a distribution is variant under a change of variable.

[9] $\log \boldsymbol{\alpha}$ is a vector obtained by applying the logarithm element-wise to vector $\boldsymbol{\alpha}$.

	prior defined as	
optimise wrt.	$p(\boldsymbol{\alpha})$ uniform	$p(\log\boldsymbol{\alpha})$ uniform
$\boldsymbol{\alpha}$	$\log p(\mathbf{y}\|X,\boldsymbol{\alpha})$	$\log p(\mathbf{y}\|X,\boldsymbol{\alpha}) + \log\boldsymbol{\alpha}$
$\log\boldsymbol{\alpha}$	$\log p(\mathbf{y}\|X,\log\boldsymbol{\alpha}) - \log\boldsymbol{\alpha}$	$\log p(\mathbf{y}\|X,\log\boldsymbol{\alpha})$

Table 2.1: Expression of the log posterior (ignoring constants) as a function of $\boldsymbol{\alpha}$ and of $\log\boldsymbol{\alpha}$, with flat priors on $\boldsymbol{\alpha}$ and on $\log\boldsymbol{\alpha}$. The MAP solution depends on changes of variable.

analytic computation of the predictive distribution. For his specific choice of independent Gamma distribution on the α's, Tipping (2001, Sect. 5.1) shows that $p_{\mathbf{w}}(\mathbf{w}) = \prod_i p_{w_i}(w_i)$ is a product of Student-t distributions, which gives a posterior over $\mathbf{w}$ that does not allow analytic integration. One option would be to again consider a MAP approach;[10] Tipping (2001, Sect. 5.1) argues that $p_{\mathbf{w}}(\mathbf{w})$ being strongly multi-modal, none of the MAP solutions is representative enough of the distribution of the (marginal) posterior probability mass. An alternative approximate inference strategy is proposed by Wipf et al. (2004), consisting in using a variational approximation to the analytically conflictive true prior. Using dual representations of convex functions, they obtain variational lower bounds on the priors on the individual weights $\tilde{p}_{\mathbf{w}}(\mathbf{w}|\mathbf{v}) = \prod_i \tilde{p}_{w_i}(w_i|v_i)$, where $v = [v_i, \ldots, v_M]$ are the variational parameters. For the special case of uniform priors over the α's,[11] the variational bounds happen to be Gaussian distributions with zero mean and variance v_i: $\tilde{p}_{w_i}(w_i) \sim \mathcal{N}(0, v_i)$. To learn $\mathbf{v}$, Wipf et al. (2004) propose to minimise the "sum of the misaligned mass":

$$
\begin{aligned}
\{v_i\} &= \arg\min_{\mathbf{v}} \int \left| p(\mathbf{y}|X,\mathbf{w},\sigma^2)\, p_{\mathbf{w}}(\mathbf{w}) - p(\mathbf{y}|X,\mathbf{w},\sigma^2)\, \tilde{p}_{\mathbf{w}}(\mathbf{w}) \right| \, \mathrm{d}\mathbf{w} \ , \\
&= \arg\max_{\mathbf{v}} \int p(\mathbf{y}|X,\mathbf{w},\sigma^2)\, \tilde{p}_{\mathbf{w}}(\mathbf{w}|\mathbf{v}) \ , \\
&= \arg\max_{\mathbf{v}} p(\mathbf{y}|X,\mathbf{v},\sigma^2) \ ,
\end{aligned}
\tag{2.36}
$$

which is equivalent to maximising the log evidence. We quickly realize that with $v_i = \alpha_i^{-1}$ this inference strategy is strictly identical to the standard RVM formulation. The difference is that the conditional priors on the weights are now the fruit of a variational approximation to the true prior on $\mathbf{w}$. For a uniform distribution over the log variance of the conditional prior, the marginal prior is given by $p_{w_i}(w_i) \propto 1/|w_i|$. Tipping (2001, Sect. 5.1) notes that this prior is analogous to the 'L_1' regulariser $\sum_i |w_i|$ and that it is reminiscent of Laplace priors $p(w_i) \propto \exp(-|w_i|)$, which have been utilised to obtain sparseness

[10] MacKay (1999) performs a comparison between integrating out parameters and optimising hyperparameters (MLII or 'evidence') and integrating over hyperparameters and maximising the 'true' posterior over parameters.

[11] A Gamma distribution $p(\alpha) \propto \alpha^{a-1} \exp(-b\,\alpha)$ tends to a uniform when $a, b \to 0$.

both in Bayesian and non-Bayesian contexts. Yet this is not enough for us to understand why the variational approximation proposed by Wipf et al. (2004) should lead to sparse solutions, and we must resort to their intuitive explanation that we reproduce in Fig. 2.4. We plot one level curve in a 2-dimensional example. The variational approximations are Gaussians confined within the marginal prior on the weights. We can see that if one of the two variational distributions had to be chosen, it would be the one for which v_1 is smaller, since it concentrates more of its mass on the likelihood. Sparsity will arise if $\mathbf{v}$ is learned: indeed v_1 will be shrunk to zero, allowing to concentrate a maximum amount of variational mass on the likelihood, which results in pruning w_1. At this point it can also intuitively be understood why sparsity only arises if the marginal prior is $p_{w_i}(w_i) \propto 1/|w_i|$, corresponding to a flat prior over the log variance of the conditional prior. If this was not the case, the spines of the marginal prior would be finite, and this would not allow the variational distributions to place infinite mass on any region, i.e. no v_i could be shrunk to zero. It is interesting to note that $p_{w_i}(w_i) \propto 1/|w_i|$ is an improper prior that cannot be accepted under a proper Bayesian treatment. Its use despite this fact would imply a rather unusual prior assumption about the model: that all weights should be zero.

Whether one considers the MAP approach with flat priors or the variational approximation to the true prior on the weights, it is the approximation made to a particular prior that enforces sparseness. Under a full Bayesian treatment, which given the impossibility of analytical integration should be accomplished via MCMC, it is difficult to see how sparseness could arise. Indeed, when integrating over the true weights posterior we do average over different sparse configurations, corresponding to different modes of the posterior. Inference, it seems, needs to be decoupled from model representation: only for the second does it seem possible to achieve sparseness, conditioned on making approximations. It is in any case incorrect in the strict sense to consider “Sparse Bayesian” models, and one should probably rather speak of “Sparse Approximate Bayesian” models.

2.3 Example: Time Series Prediction with Adaptive Basis Functions

In (Quiñonero-Candela and Hansen, 2002) we used the RVM for time series prediction. We chose a hard prediction problem, the MacKey-Glass chaotic time series, which is well-known for its strong non-linearity. Optimised non-linear models can have a prediction error which is three orders of magnitude lower than an optimised linear model (Svarer et al., 1993). The Mackey-Glass

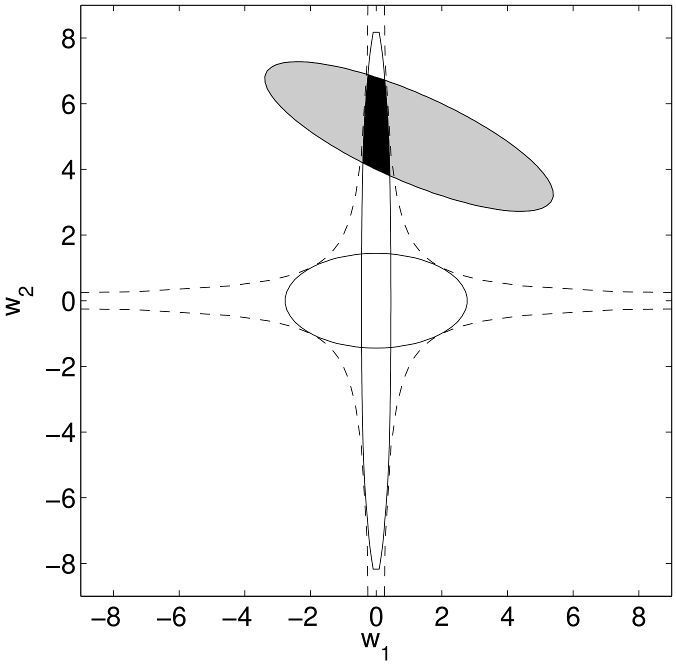

Figure 2.4: Contour plots to understand sparsity. The dashed lines depict the true prior $p_{\mathbf{w}}(\mathbf{w}) \propto 1/(|w_1| \cdot |w_2|)$. The gray filled Gaussian contour represents the likelihood $p(\mathbf{y}|X, \mathbf{w}, \sigma^2)$. The two empty Gaussian contours represent two candidate variational approximations to the posterior $\tilde{p}_{\mathbf{w}}(\mathbf{w}) \sim \mathcal{N}(0, \mathrm{diag}(\mathbf{v}))$: the vertically more elongated one will be chosen, since it concentrates more mass on areas of high likelihood. If $\mathbf{v}$ is learned, v_1 will be shrunk to zero to maximise the variational mass on the likelihood. This will prune w_1: sparsity arises.

attractor is a non-linear chaotic system described by the following equation:

$$\frac{dz(t)}{dt} = -bz(t) + a\frac{z(t-\tau)}{1 + z(t-\tau)^{10}} \tag{2.37}$$

where the constants are set to $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is re-sampled with period 1 according to standard practice. The inputs are formed by $L = 16$ samples spaced 6 periods from each other $\mathbf{x}_k = [z(k-6), z(k-12), \ldots, z(k-6L)]$ and the targets are chosen to be $y_k = z(k)$ to perform six steps ahead prediction (Svarer et al., 1993, as in). The fact that this dataset has virtually no output noise, combined with its chaotic nature prevents sparsity from arising in a trivial way: there is never too much training data. Fig. 2.5 (left) shows 500 samples of the chaotic time series on which we train an RVM.

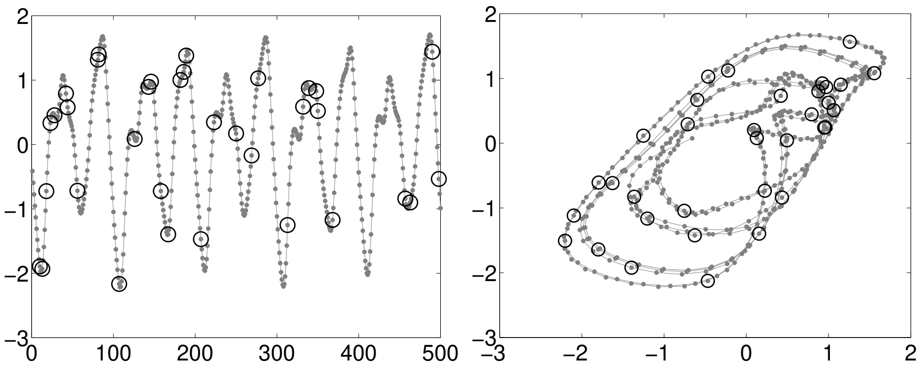

Figure 2.5: Sparse RVM model: 34 RVs are selected out of 500 basis functions. *Left*: targets $y_k = z(k)$ corresponding to the selected RVs. (*Left*): location of the RVs in the 2-dimensional space span by the two first components of $\mathbf{x}$, $[z(k-6), z(k-12)]$.

Only 34 RVs are retained: the targets corresponding to the RVs are shown on the left pane, and the inputs corresponding to the RVs are shown on the right pane, in a reduced 2-dimensional input space where $\mathbf{x} = [z(k-6), z(k-12)]$. In this example we have used an RVM with squared exponential basis functions, and this time we have also *learned* the lengthscales, one for each input dimension individually. The test mean square error of the resulting model was 3×10^{-4} on a test set with 6000 elements.

The RVM can have very good performance on the Mackey-Glass time series, compared to other methods. Yet this performance depends heavily on the choice of the lengthscales of the basis functions. We show this effect for isotropic basis functions in Fig. 2.6, reproduced from (Quiñonero-Candela and Hansen, 2002). We train on 10 disjoint sets of 700 elements and test on sets of 8500 elements. On the one hand we train an RVM with fixed lengthscale of value equal to the initial value given in the figure. On the other hand we train an RVM and adapt the lengthscale from an initial value given in the horizontal axis. We present the test mean square error for both models and for each initial value of the lengthscale. It can be seen that the performance dramatically depends on the lengthscale. On the other hand, the experiment shows that it is possible to learn it from a wide range of initial values. In (Quiñonero-Candela and Hansen, 2002) we also observed that the number of RVs retained is smaller, the larger the lengthscale (in Fig. 2.2 we observed that for fixed lengthscales, the number of RVs is smaller the larger the estimated output noise).

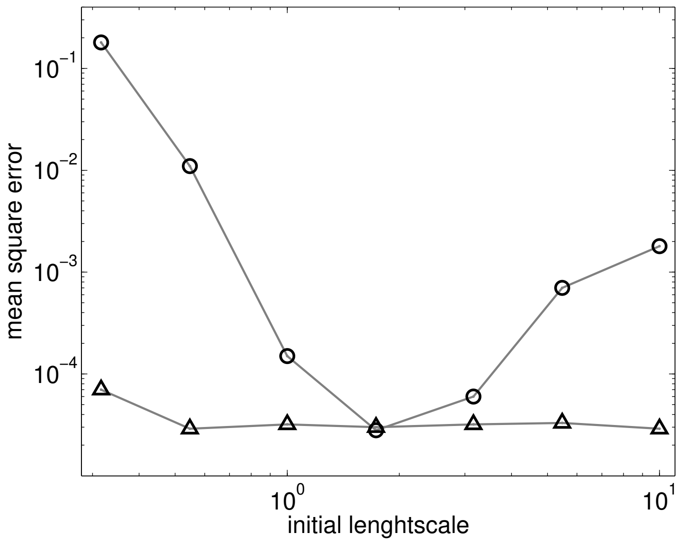

Figure 2.6: Test mean square error on with and without adapting the variance of the basis functions for an RVM with squared exponential basis functions. Averages over 10 repetitions are shown, training on 500 and testing on 6000 samples of the Mackey-Glass time-series. The horizontal axis shows the value to which the lengthscale is initialised: the *triangles* show the test mean square error achieved by adapting the lengthscales, and the *circles* correspond to the errors achieved with the fixed initial values of the lengthscales.

2.3.1 Adapting the Basis Functions

In (Quiñonero-Candela and Hansen, 2002) we learned the lengthscale of the isotropic basis functions by maximising the log evidence. We used a simple direct search algorithm (Hooke and Jeeves, 1961) at the M-step of the modified EM algorithm (2.32). Direct search was possible on this simple 1-dimensional problem, but one may be interested in the general case where one lengthscale is assigned to each input dimension; this would in general be a too high dimensional problem to be solved by direct search. Tipping (2001, App. C) derives the derivatives of the log evidence wrt. to general parameters of the basis functions. For the particular case of squared exponential basis functions, of the form given by (2.6), it is convenient to optimise with respect to the logarithm of the lengthscales:

$$\frac{\partial \mathcal{L}}{\partial \log \theta_d} = - \sum_{i=1}^{N} \sum_{j=1}^{M} \mathbf{D}_{ij} \, \phi_j(\mathbf{x}_i) \, (X_{id} - X_{jd})^2 \ . \tag{2.38}$$

method	*train error*	*test error*
Simple Linear Model	9.7×10^{-2}	9.6×10^{-2}
5NN Linear Model	4.8×10^{-7}	8.4×10^{-5}
Pruned MLP	3.1×10^{-5}	3.4×10^{-5}
RVM Isotropic	2.3×10^{-6}	5.5×10^{-6}
RVM Non-Isotropic	1.1×10^{-6}	1.9×10^{-6}

Table 2.2: Training and test mean square prediction error for the Mackey-Glass chaotic time series. Averages over 10 repetitions, 1000 training, 8500 test cases. Models compared (top to bottom): simple linear model on the inputs, 5 nearest neighbours local linear model on the inputs, pruned multilayer perceptron, RVM with adaptive isotropic squared exponential basis functions, and the same with individual lengthscales for each input dimension.

where $\mathbf{D} = \left[(\mathbf{y} - \mathbf{\Phi}\,\boldsymbol{\mu})\,\boldsymbol{\mu}^\top - \mathbf{\Phi}\,\mathbf{\Sigma}\right]$. The cost of computing these derivatives is $\mathcal{O}(NM^2 + NMD)$, while the cost of computing the log evidence is $\mathcal{O}(NM^2)$, since Φ varies with θ_d, and $\Phi^\top\Phi$ needs to be recomputed. Roughly, for the isotropic case, as long as direct search needs less than twice as many function evaluations as gradient descent, it is computationally cheaper.

For completeness, we extend here the experiments we performed in (Quiñonero-Candela and Hansen, 2002) to the case of non-isotropic squared exponential basis functions. The results are given in Table 2.2. In those experiments, we compared an RVM with adaptive isotropic, lengthscales with a simple linear model, with a 5 nearest-neighbours local linear model and with the pruned neural network used in Svarer et al. (1993) for 6 steps ahead prediction. The training set contains 1000 examples, and the test set 8500 examples. Average values of 10 repetitions are presented. The RVM uses an average of 108 RVs in the isotropic case, and an average of 87 for the non-isotropic case. It is remarkable that the Adaptive RVM so clearly outperforms an MLP that was carefully optimised by Svarer et al. (1993) for this problem. It is also remarkable that much improvement can be gained by individually learning the lengthscales for each dimension. Compared to the isotropic case, sparser models are obtained, which perform better!

Unfortunately, the success of optimising the lengthscales critically depends on the way the optimisation is performed. Specifically, a gradient descent algorithm is for example used at the M-step to update the values of the lengthscales by maximising the log evidence. The ratio between the amount of effort put into optimising α and σ^2 and the amount of effort put into optimising the θ_d's critically determines the solution obtained. Tipping (2001, App. C) mentions that "the exact quality of results is somewhat dependent of the ratio of the number of $\boldsymbol{\alpha}$ to η updates" (he refers to η as the inverse squared lengthscales). We have for example encountered the situation in which optimising the lengthscales too heavily in the initial stages, where all α's are small, leads to getting stuck with too small lengthscales that lead to over-fitting. Joint optimisation of the lengthscales does not seem a trivial task, and careful tuning is required to obtain satisfactory results. For the concrete case of the Mackey-Glass time-series predictions with individual lengthscales, we have found that performing a partial conjugate gradient ascent (with only 2 line searches) at each M-step gives good results.

2.4 Incremental Training of RVMs

Until now the computational cost of training an RVM has not been addressed in this chapter. Yet this is an important limiting factor for its use on large training datasets. The computational cost is marked by the need of inverting $\boldsymbol{\Sigma}$, at a cost of $\mathcal{O}(M^3)$, and by the computation of $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$, at a cost of $\mathcal{O}(NM^2)$.[12] Initially, before any of the α's grows to infinity, we have that $M = N$ (or $M = N + 1$, if a bias basis function is added). This implies a computational cost cubic in the number of training examples, which makes training on datasets with more than a couple thousand examples impractical. The memory requirements are $\mathcal{O}(N^2)$ and can also be limiting.

Inspired by the observation of Tipping (2001, App. B.2) that the RVM could be trained in a "constructive" manner, in (Quiñonero-Candela and Winther, 2003) we proposed the Subspace EM (SSEM) algorithm, an incremental version of the EM (or of the faster approximate EM) algorithm used for training RVMs presented in Sect. 2.2.1. The idea is to perform the E and M-steps only in a subset of the weight space, the *active set*. This active set is iteratively grown, starting from the empty set. Specifically, at iteration n the active set R_n contains the indices of the α's who are allowed to be finite. As opposed to the standard way of training RVMs, the active set is initially empty, corresponding to a fully pruned model. The model is grown by iteratively including in the active set the index of some weight, selected at random from the indices of the α's set to infinity. After each new inclusion, the standard EM for the RVM is run on the active set only, and some α's with index in the active set may be set to infinity again (and become again candidates for inclusion). This procedure is equivalent to iteratively presenting a new basis function to the model, and letting it readjust its parameters to decide whether it incorporates the new basis function and whether it prunes some older basis function in the light of the newly

[12] For fixed basis functions, i.e. if the lengthscales are not learned, $\boldsymbol{\Phi}^\top \boldsymbol{\Phi}$ can be precomputed at the start, eliminating this cost at each iteration.

1. Set $\alpha_j = \mathsf{L}$ for all j. (L is effectively infinity) Set $n = 1$
2. Update the set of active indexes R_n
3. Perform an E-step in subspace ω_j such that $j \in R_n$
4. Perform the M-step for all α_j such that $j \in R_n$
5. If all basis functions have been visited, end, else go to 2.

Figure 2.7: Schematics of the SSEM algorithm.

acquired basis function. Given the active set at step $n-1$, the active set at step n is given by:

$$R_n = \{i \mid i \in R_{n-1} \wedge \alpha_i \leq +\infty\} \cup \{n\} \quad , \tag{2.39}$$

where of course $+\infty$ is in practice a very large finite number L arbitrarily defined. Observe that R_n contains at most one more element (index) than R_{n-1}. If some of the α's indexed by R_{n-1} happen to reach L at the n-th step, R_n can contain less elements than R_{n-1}. This implies that R_n contains at most n elements, and typically less because of pruning. In Fig. 2.7 we give a schematic description of the SSEM algorithm.

At step n, we want to maximise the marginal likelihood with respect to the noise and to the α_j such that $j \in R_n$, while the remaining are treated as constants with infinite value. The E-step corresponds to computing the mean and the covariance of a reduced posterior on the weights with index in R_n. The expressions (2.19) and (2.18) can directly be used, where $\mathbf{\Phi}$ now has one row for each index in R_n, and $\mathbf{A}$ is the diagonal matrix with elements α_j with $j \in R_n$. The M-step corresponds to re-estimating the α's in the active set by plugging the estimate of the reduced posterior in (2.30) if the exact EM is used, or in (2.32) if the faster, approximate EM is used instead. At the n-th iteration, the computational complexity of the SSEM algorithm is bounded from above by $\mathcal{O}(n^3)$. In practice we have observed that this complexity is smaller, since at iteration n the size of the active set is significantly smaller than n due to pruning.

It must be noted that since the initial value of α_j is infinity for all j, the E-step yields always an equality between the log marginal likelihood and its lower bound. At any step n, the posterior can be exactly projected on to the space spanned by the weights $\mathbf{w}_j$ with $j \in R_n$. Hence, if used with the exact M-step, the SSEM never decreases the log evidence. A subtlety needs to be addressed, relative to the exact manner in which the SSEM algorithm is used. After the update of the active set, one must decide how many EM iterations to perform before updating the active set again. The one extreme is to perform only one EM update, which is what we did in (Quiñonero-Candela and Winther (2003)).

The other extreme is to perform a "full" EM until some convergence criterion is satisfied. An additional consideration is that a straightforward modification of the SSEM algorithm is to include more than one index to the active set at each iteration. We are then left with two degrees of freedom: the number of EM cycles between updates of the active set, and the amount by which the active set is increased when updated. This flexibility can be used to set a limit to the maximum number of elements in the active set. This number can be kept constant, by incrementing the active set by the number of weights pruned at each EM iteration. In this way one can choose the amount of computational effort that one wants to put into the learning process. In Fig. 2.8 we show the evolution of the negative log likelihood and the number of RVs for an incremental RVM where the size of the active set is fixed to 500 (SSEM-fixed-500) and for an incremental RVM where the active set is incremented with 59 elements[13] every 10 EM cycles (SSEM-10-59). The two are compared to the standard RVM on an 8-dimensional training set with 4000 training examples, the KIN40K dataset, that we describe in Sect. 2.6. The stopping criterion is based on the relative change of the negative log likelihood; when it is met, an additional 100 EM cycles are performed with no new inclusions in the active set. SSEM-fixed-500 achieves a better negative log evidence, and also a slightly better test error (0.037 versus 0.039, although this might not be significant) than SSEM-59. The final number of RVs selected is similar for the three methods, 340 for the RVM, 372 for SSEM-59 and 390 for SSEM-fixed-500. A significant computational speedup is gained by using the SSEM algorithm, already for a training set of 4000 examples. For small enough training sets, it is preferable to directly use standard RVM, which faster than SSEM. Conversely, the larger the training set, the greater the speedup offered by SSEM. SSEM also allows to train on very large training sets (sizes of 10000 or more) where the standard RVM cannot be used at all (on the computers within our reach).

Tipping and Faul (2003) have recently proposed an alternative incremental approach to training the RVM that takes advantage of the analysis performed by Faul and Tipping (2002), where the maximum of the log evidence wrt. to a single α can be computed in closed-form. The "fast marginal likelihood maximisation" they propose selects the next basis function to include in the active set under an optimality criterion, which one would expect to be better than random selection. However, this active selection process comes at an additional computational cost. It will be interesting to perform an exhaustive practical comparison between the two methods, to learn whether the method proposed by Tipping and Faul (2003) is superior on all fronts, or whether for applications where minimal computational complexity is required our method offers sufficiently good performance. It will also be interesting to investigate the com-

[13] "Magic" number used by Smola and Bartlett (2001), that gives a 0.95 probability of including one among the 0.05 best basis functions.

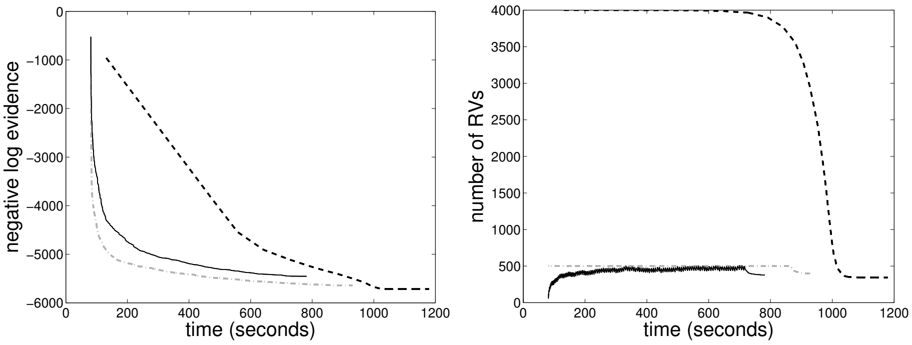

Figure 2.8: Negative log evidence (*left*) and number of RVs (*right*) versus training time (in seconds). The dashed line corresponds to a standard RVM, the gray dash-dotted line to the SSEM training with fixed active set size of 5000, and the black solid line to the SSEM training with increments of 59 basis functions every 10 EM cycles. The dataset is 8-dimensional and contains 4000 training examples.

parative performance of an alternative efficient approach to training RVMS, based on Bayesian "backfitting" rather than on an incremental scheme, that was very recently proposed by D'Souza et al. (2004).

2.5 Improving the Predictive Variances: RVM*

Probabilistic models are interesting because instead of point predictions they provide with predictive distributions. The extended linear models with Gaussian priors on the weights we have considered in this chapter are probabilistic models, that have simple Gaussian predictive distributions with mean and variance given by (2.10). For the RVM, this is also the case since the true priors on the weights are approximated by Gaussian priors conditioned on the α's. For localised basis functions such as the squared exponential given by (2.6), if a test input $\mathbf{x}_*$ lies far away from the centres of all relevance vectors, the response of the basis functions $\phi_j(\mathbf{x}_*)$ becomes small: the predictive mean goes to zero and the predictive variance reduces to the noise level. Under the prior, functions randomly generated from (2.1) will not have much signal far away from the centres of the relevance vectors. In other words, the model uncertainty is maximal in the neighbourhood of the centres of the relevance vectors, and goes to zero as one moves far away from them. For the posterior, this effect is illustrated in Fig. 2.9, and can be further understood by analysing the expression of the

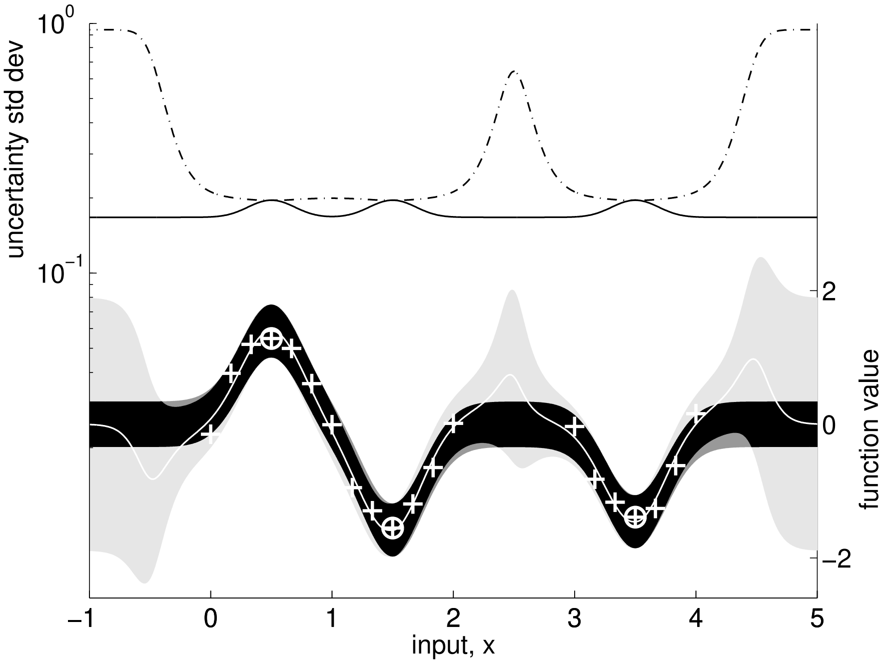

Figure 2.9: Behaviour of the predictive distributions. *Bottom* (with left y-axis): the white crosses represent the training data, the circled ones being the cases whose input is the center of the relevance vectors obtained from training. The black region is the 95% confidence interval for the predictions of a standard RVM, and the gray region that for the RVM* (for the latter, the white line is the mean prediction). *Top* (with right y-axis): The solid line represents the predictive standard deviation of the RVM, and the dot-slashed one that of the RVM*. Note that the predictive variance *decreases* when moving away from the relevance vectors in the RVM, but *increases* for the RVM*.

predictive variance, (2.10).

As a probabilistic model, the RVM with localised basis functions thus produces inappropriate estimates of the predictive uncertainty, with a behaviour opposite to what would seem desirable.

2.5.1 RVM*

Consider having trained an RVM and facing the task of predicting at a new unseen test point. To solve the problem, that there might be no possibility of variation for inputs far from the centres of the relevance vectors, we propose the introduction of an additional basis function centred at the test point. The training stage remains *unchanged*, the new basis function being introduced only at test time and for a specific test input. This is the idea behind the modification of the RVM that we propose: the RVM*.

For each test point $\mathbf{x}_*$, we modify the model obtained from training by introducing one new basis function centred on $\mathbf{x}_*$, and its associated weight with prior distribution $p(w^*) \sim \mathcal{N}(0, \alpha_*^{-1})$ (we postpone a little the issue of how to set α_*). The joint augmented posterior distribution of the weights has now covariance and mean given by:

$$\boldsymbol{\Sigma}_* = \begin{bmatrix} \boldsymbol{\Sigma}^{-1} & \sigma^{-2}\boldsymbol{\Phi}^\top \boldsymbol{\phi}_* \\ \sigma^{-2}\boldsymbol{\phi}_*^\top \boldsymbol{\Phi} & \alpha_* + \sigma^{-2}\boldsymbol{\phi}_*^\top \boldsymbol{\phi}_* \end{bmatrix}^{-1} , \qquad \boldsymbol{\mu}_* = \sigma^{-2}\boldsymbol{\Sigma}_* \begin{bmatrix} \boldsymbol{\Phi}^\top \\ \boldsymbol{\phi}_*^\top \end{bmatrix} \mathbf{y} . \tag{2.40}$$

$\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ are the covariance and the mean of the posterior distribution of the weights obtained from training, (2.18) and (2.19), and ϕ_* is the newly introduced basis function evaluated at all training inputs.

The predictive distribution of the augmented model at $\mathbf{x}_*$ has mean and variance given by:

$$m_*(\mathbf{x}_*) = m(\mathbf{x}_*) + \frac{e_*\, q_*}{\alpha_* + s_*} , \qquad v_*(\mathbf{x}_*) = v(\mathbf{x}_*) + \frac{e_*^2}{\alpha_* + s_*} , \tag{2.41}$$

where

$$q_* = \boldsymbol{\phi}_*^\top(\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\mu})/\sigma^2 , \qquad s_* = \boldsymbol{\phi}_*^\top(\sigma^2\mathbf{I} + \boldsymbol{\Phi}A^{-1}\boldsymbol{\Phi}^\top)^{-1}\boldsymbol{\phi}_* ,$$
$$e_* = 1 - \sigma^{-2}\phi(\mathbf{x}_*)\boldsymbol{\Sigma}\,\boldsymbol{\Phi}^\top\boldsymbol{\phi}_* .$$

We have adopted the notation of Faul and Tipping (2002): q_* is a 'quality' factor, that indicates how much the training error can be reduced by making use of the new basis function. s_* is a 'sparsity' factor that indicates how much the new basis function is redundant for predicting the training data given the existing relevance vectors. e_* is an 'error' term, that indicates how much worse the existing model is than the new basis function at predicting at the new input $\mathbf{x}_*$. Note, that the predictive variance of RVM* in eq. (2.41) is guaranteed not to be smaller than for the RVM. Note also, that the predictive mean of the RVM* is modified as a result of the additional modelling flexibility, given by the new basis function. This new basis function is weighted according to how much it helps model the part of the training data that was not well modelled by the classic RVM, whose sparseness may lead to under-fitting, see the discussion section. Figure 2.9 illustrates this effect.

When introducing an additional basis function at test time, we also get an additional weight w_* (which is integrated out when making predictions) and an extra prior precision parameter α_*. How do we set α_*? One naïve approach would be to take advantage of the work on incremental training done by Faul and Tipping (2002), where it is shown that the value of α_* that maximizes the marginal likelihood, given all the other α's (in our case obtained from training) is given by:

$$\alpha_* = \frac{s_*^2}{q_*^2 - s_*}, \quad \text{if } q_*^2 > s_*, \qquad \alpha_* = \infty, \quad \text{otherwise.} \tag{2.42}$$

Unfortunately, this strategy poses the risk of deletion of the new basis function (when the new basis function doesn't help significantly with modelling the data, which is typically the case when when $\mathbf{x}_*$ lies far from all the training inputs). Thus the unjustifiably small error bars of RVM would persist.

In our setting learning α_* by maximizing the evidence makes little sense, since it contravenes the nature of our approach. We do want to impose an a priori assumption on the variation of the function. When far away from the relevance vectors, α_*^{-1} is the a priori variance of the function value. We find it natural to make α_*^{-1} equal to the empirical variance of the observed target values, corresponding to the prior assumption that the function may vary everywhere.

Training is identical for the RVM and for the RVM*, so it has the same computational complexity for both. For predicting, the RVM needs only to retain $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ from training, and the complexity is $\mathcal{O}(M)$ for computing the predictive mean and $\mathcal{O}(M^2)$ for the predictive variance. The RVM* needs to retain the whole training set in addition to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. The computational complexity is $\mathcal{O}(MN)$ both for computing the predictive mean and the predictive variance. The dependence on the full training set size N is caused by the additional weight needing access to all targets[14] for marginalized over.

[14]One could get rid of the dependence on N by re-fitting only using the targets associated with the relevance vectors; this leads to too large predictive variances, since the training set may have contained data close to the test input, which hadn't been designated as relevance vectors.

	Squared error loss			Absolute error loss			- log test density loss		
	RVM	RVM*	GP	RVM	RVM*	GP	RVM	RVM*	GP
Loss:	0.138	0.135	0.092	0.259	0.253	0.209	0.469	0.408	0.219
RVM	·	not sig.	< 0.01	·	0.07	< 0.01	·	< 0.01	< 0.01
RVM*		·	0.02		·	< 0.01		·	< 0.01
GP			·			·			·

Table 2.3: Results for the Boston house-price experiments for RVM, RVM* and GP. The upper sub-table indicates the average value of the losses for three loss functions. In the lower sub-table, the values in the cells are the p-values that indicate the significance with which the model in the corresponding column beats the model in the corresponding row.

2.6 Experiments

We compare the classic RVM, the RVM* and a Gaussian process (GP)[15] with squared exponential covariance function on two datasets: the Boston house-price dataset, (Harrison and Rubinfeld, 1978), with 13-dimensional inputs, and the KIN40K (robot arm) dataset[16], with 8-dimensional inputs. The KIN40K dataset represents the forward dynamics of an 8 link all-revolute robot arm.

We use a 10 fold cross-validation setup for testing on both datasets. For Boston house-price we use disjoint test sets of 50/51 cases, and training sets of the remaining 455/456 cases. For the robot arm we use disjoint test and training sets both of 2000 cases. For all models we learn individual lengthscales for each input dimension, and optimize by maximizing the marginal likelihood, (Tipping, 2001, appendix C) and (Williams and Rasmussen, 1996). For each partition and model we compute the squared error loss, the absolute error loss and the negative log test density loss. In addition to the average values of the different losses, we compute the statistical significance of the difference in performance of each pair of models for each loss, and provide the p-value obtained from a (two-sided) paired t-test[17] on the test set averages.

The results for the Boston house-price example in table 2.3 show that the RVM* produces significantly better predictive distributions than the classic RVM. Whereas the losses which only depend on the predictive mean (squared and absolute) are not statistically significantly different between RVM an RVM*,

[15] We give an introduction to Gaussian Processes in Sect. 3.1.

[16] From the DELVE archive http://www.cs.toronto.edu/delve.

[17] For the Boston house-price dataset, due to dependencies (overlap) between the training sets, assumptions of independence needed for the t-test are compromised, but this is probably of minor effect.

	Squared error loss			Absolute error loss			- log test density loss		
	RVM	RVM*	GP	RVM	RVM*	GP	RVM	RVM*	GP
Loss:	0.0043	0.0040	0.0024	0.0482	0.0467	0.0334	-1.2162	-1.3295	-1.7446
RVM	·	< 0.01	< 0.01	·	< 0.01	< 0.01	·	< 0.01	< 0.01
RVM*		·	< 0.01		·	< 0.01		·	< 0.01
GP			·			·			·

Table 2.4: Results for the Robot Arm data; the table is read analogously to table 2.3.

the negative log test density loss is significantly smaller for RVM*, confirming that the predictive uncertainties are much better. The RVM models have a final average number of relevance vectors of 27 ± 14 (mean $\pm$ std. dev.) showing a high degree of sparsity[18] and quite some variability. The results for the Robot arm example in table 2.4 show a similar picture. For this (larger) data set, the difference between RVM and RVM* is statistically significant even for the losses only depending on the mean predictions. The final numbers of relevance vectors were 252 ± 11. We also compare to a non-degenerate (see section 2.6.1) Gaussian process. The GP has a significantly superior performance under all losses considered. Note also, that the difference between RVM and GPs is much larger than that of RVM vs. RVM*. This may indicate that sparsity in regression models may come at a significant cost in accuracy. To our knowledge, RVMs and GPs have not been compared previously experimentally in an extensive manner.

2.6.1 Discussion

The RVM is equivalent to a GP (Tipping, 2001, section 5.2) with covariance function given by:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{M} \frac{1}{\alpha_k} \phi_k(\mathbf{x}_i)\, \phi_k(\mathbf{x}_j). \tag{2.43}$$

This covariance function is *degenerate*,[19] in that when $M < N$ (which is typical for the RVM) the distribution over (noise free) functions is singular. This limits the range of functions that can be implemented by the model. The RVM* introduced in this paper is a GP with an augmented covariance function:

$$k_*(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{\alpha_*} \phi_*(\mathbf{x}_i)\, \phi_*(\mathbf{x}_j), \tag{2.44}$$

[18] Note, that the degree of sparsity obtained depends on the (squared exponential) basis function widths; here the widths were optimized using the marginal likelihood.

[19] In Sect. 3.2.3 we elaborate more on degenerate GPs.

which ensures prior variability at the test location, that survives into the posterior if the data doesn't have a strong opinion in that region.

It is interesting to note that a GP with squared exponential covariance function coincides exactly with an RVM infinitely augmented, at all points in the input space. Following Mackay (1997), in Sect. 3.2.1 we show this for the one-dimensional input case, where we recover the squared exponential covariance GP as being equivalent to an infinite RVM. The infinite RVM becomes tractable when viewed as a GP, but of course it is not clear how to treat the infinitely many hyperparameters, or how to introduce sparsification from this standpoint.

It may be surprising that the experiments show that the performance using loss functions which depend only on the predictive means was improved for the RVM* (although sometimes the difference was not statistically significant). The reason for this is that the extra added basis function, which is fit to the training data, adds flexibility to the model. Since this extra flexibility turns out to improve performance, this shows that the classical RVM under-fits the data, ie. the models have become too sparse. Indeed the performance of the full non-degenerate GP is much better still.

The RVM has become a popular tool because it represents a simple tractable probabilistic model. As we have shown, if one is interested in the predictive variance, the RVM should not be used. Even if one is interested only in predictive means, the sparsity mechanism of the RVM seems to come at the expense of accuracy. The proposed RVM* goes some way at fixing this problem at an increased computational cost. Although outside the scope of this thesis, it is an important future task to experimentally compare the computation vs. accuracy tradeoff between different methods for sparsifying GPs. Some recent papers do attempt to assess these tradeoffs, however, the performance measures often neglect the probabilistic nature of the predictions and focus exclusively on mean predictions. In Chap. 3 we investigate one family of sparse approximations to GPs.

CHAPTER 3

Reduced Rank Gaussian Processes

Gaussian Processes (GPs) are known for their state of the art performance in regression and classification problems. In this chapter we restrict ourselves to the regression case, and introduce GPs in Sect. 3.1. Relevance Vector Machines (RVMs), that we presented in Chap. 2, are in fact particular examples of GPs. In Sect. 3.2 we establish the more general fact that GPs are equivalent to extended linear models when Gaussian priors are imposed over the weights, and that the number of weights can be infinite. The linear model perspective is later used to devise computationally effective approximations to GPs.

Indeed, GPs unfortunately suffer from high computational cost for learning and predictions. For a training set containing N cases, the complexity of training is $\mathcal{O}(N^3)$, similar to that of the RVM (Sect. 2.2), and that of making a prediction is $\mathcal{O}(N)$ for computing the predictive mean, and $\mathcal{O}(N^2)$ for computing the predictive variance (versus $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ for the RVM, with $M \ll N$). A few computationally effective approximations to GPs have recently been proposed. Sparsity is achieved in Csató (2002), Csató and Opper (2002), Seeger (2003), and Lawrence et al. (2003), by minimising KL divergences between the approximated and true posterior; Smola and Schölkopf (2000) and Smola and Bartlett (2001) based on a low rank approximate posterior; Gibbs and MacKay (1997) and Williams and Seeger (2001) using matrix approximations; Tresp (2000) by neglecting correlations; and (Wahba et al., 1999; Poggio and Girosi, 1990) with subsets of regressors. The RVM (Tipping, 2001) can also be cast as a sparse linear approximation to GPs, although it was not conceived as such. Schwaighofer and Tresp (2003) provide a very interesting yet brief comparison of some of these approximations to GPs. They only address the quality of the approximations in terms of the predictive mean, ignoring the predictive uncertainties, and leave some theoretical questions unanswered, like the validity of approximating the the maximum of the posterior.

In Sect. 3.3.1 we analyse sparse linear or equivalently reduced rank approximations to GPs that we will call Reduced Rank Gaussian Processes (RRGPs). In a similar manner to the RVM, RRGPs correspond to inappropriate priors over functions, resulting in inappropriate predictive variances (for example, the predictive variance shrinks as the test points move far from the training set). We give a solution to this problem which is analogous to the RVM* proposed in Sect. 2.5, consisting in augmenting the finite linear model at test time. This guarantees that the RRGP approach corresponds to an appropriate prior. Our analysis of RRGPs should be of interest in general for better understanding the infinite nature of Gaussian Processes and the limitations of diverse approximations (in particular of those based solely on the posterior distribution). Learning RRGPs implies both selecting a support set, and learning the hyperparameters of the covariance function. Doing both simultaneously proves to be difficult in practice and questionable theoretically. Smola and Bartlett (2001) proposed the Sparse Greedy Gaussian Process (SGGP), a method for learning the support set for given hyperparameters of the covariance function based on approximating the posterior. We show that approximating the posterior is unsatisfactory, since it fails to guarantee generalisation, and propose a theoretically more sound greedy algorithm for support set selection based on maximising the marginal likelihood. We show that the SGGP relates to our method in that approximating the posterior reduces to partially maximising the marginal likelihood. We illustrate our analysis with an example. We propose an approach for learning the hyperparameters of the covariance function of RRGPs for a given support set, originally introduced by Rasmussen (2002).

In Sect. 3.4 we present experiments where we compare learning based on selecting the support set to learning based on inferring the hyperparameters. We give special importance to evaluating the quality of the different approximations when computing predictive variances. A discussion in Sect.3.5 concludes this chapter.

3.1 Introduction to Gaussian Processes

In inference with parametric models prior distributions are often imposed over the model parameters, which can be seen as a means of imposing regularity and improving generalisation. The form of the parametric model, together with the form of the prior distribution on the parameters result in a (often implicit) prior assumption on the joint distribution of the function values. At prediction time the quality of the predictive uncertainty will depend on the prior over functions. Unfortunately, for probabilistic parametric models this prior is defined in an indirect way, and this in many cases results in priors with undesired properties. An example of a model with a peculiar prior over functions is the RVM, for which the predictive variance shrinks for a query point far away from the training inputs as discussed in Sect. 2.5. If this property of the predictive variance is undesired, then one concludes that the prior over functions was undesirable in the first place, and one would have been happy to be able to directly define a prior over functions.

Gaussian Processes (GPs) are non-parametric models where a Gaussian process[1] prior is directly defined over function values. The direct use of Gaussian Processes as priors over functions was motivated by Neal (1996) as he was studying priors over weights for artificial neural networks. A model equivalent to GPs, *kriging*, has since long been used for analysis of spatial data in Geostatistics (Cressie, 1993). In a more formal way, in a GP the function outputs $f(\mathbf{x}_i)$ are a collection random variables indexed by the inputs $\mathbf{x}_i$. Any finite subset of outputs has a joint multivariate Gaussian distribution (for an introduction on GPs, and thorough comparison with Neural Networks see (Rasmussen, 1996)). Given a set of training inputs $\{\mathbf{x}_i | i = 1, \dots, N\} \subset \mathbb{R}^D$ (organised as rows in matrix X), the joint prior distribution of the corresponding function outputs $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^\top$ is Gaussian $p(\mathbf{f}|X, \theta) \sim \mathcal{N}(0, \mathbf{K})$, with zero mean (this is a common and arbitrary choice) and *covariance matrix* $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. The GP is entirely determined by the *covariance function* $K(\mathbf{x}_i, \mathbf{x}_j)$ with parameters θ.

An example of covariance function that is very commonly used is the squared exponential:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \theta_{D+1}^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{1}{\theta_d^2} \left(X_{id} - X_{jd}\right)^2 \right) . \tag{3.1}$$

θ_{D+1} relates to the amplitude of the functions generated by the GP, and θ_d is

[1] We will use the expression “Gaussian Process” (both with capital first letter) or “GP” to designate the non-parametric model where a Gaussian process prior is defined over function values

a lengthscale in the d-th dimension that allows for Automatic Relevance Determination (ARD) (MacKay, 1994; Neal, 1996): if some input dimensions are un-informative about the covariance between observed training targets, their associated θ_d will be made large (or effectively infinite) and the corresponding input dimension will be effectively pruned from the model. We will call the parameters of the covariance function *hyperparameters*, since they are the parameters of the prior.

In general, inference requires choosing a parametric form of the covariance function, and either estimating the corresponding parameters θ (which is named by some Maximum Likelihood II, or second level of inference) or integrating them out (often through MCMC). We will make the common assumption of Gaussian independent identically distributed output noise, of variance σ^2. The training outputs $\mathbf{y} = [y_1, \ldots, y_N]^\top$ (or targets) are thus related to the function evaluated at the training inputs by a likelihood distribution[2] $p(\mathbf{y}|\mathbf{f}, \sigma^2) \sim \mathcal{N}(\mathbf{f}, \sigma^2\,\mathbf{I})$, where $\mathbf{I}$ is the identity matrix. The posterior distribution over function values is useful for making predictions. It is obtained by applying 'Bayes' rule:[3]

$$\begin{aligned} p(\mathbf{f}|\mathbf{y}, X, \theta, \sigma^2) &= \frac{p(\mathbf{y}|\mathbf{f}, \sigma^2)\, p(\mathbf{f}|X, \theta)}{p(\mathbf{y}|X, \theta, \sigma^2)} \\ &\sim \mathcal{N}\left(\mathbf{K}^\top \left(\mathbf{K} + \sigma^2\,\mathbf{I}\right)^{-1} \mathbf{y}, \mathbf{K} - \mathbf{K}^\top \left(\mathbf{K} + \sigma^2\,\mathbf{I}\right)^{-1} \mathbf{K}\right) \ . \end{aligned} \tag{3.2}$$

The mean of the posterior does not need to coincide with the training targets. This would be the case however, if the estimated noise variance happened to be zero, in which case the posterior at the training cases would be a delta function centred on the targets.

Consider now that we observe a new input $\mathbf{x}_*$ and would like to know the distribution of $f(\mathbf{x}_*)$ (that we will write as f_* for convenience) conditioned on the observed data, and on a particular value of the hyperparameters and of the output noise variance. The first thing to do is to write the augmented prior over the function values at the training inputs and the new function value at the new test input:

$$p\left(\left[\begin{array}{c} \mathbf{f} \\ f_* \end{array}\right] \middle| \mathbf{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, \left[\begin{array}{cc} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**} \end{array}\right]\right) \ , \tag{3.3}$$

where $\mathbf{k}_* = [K(\mathbf{x}_*, \mathbf{x}_1), \ldots, K(\mathbf{x}_*, \mathbf{x}_N)]^\top$ and $k_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$. Then we can write the distribution of f_* conditioned on the training function outputs:

$$p(f_*|\mathbf{f}, \mathbf{x}_*, X, \theta) \sim \mathcal{N}\left(\mathbf{k}*^\top\, \mathbf{K}^{-1}\mathbf{f}, k_{**} - \mathbf{k}*^\top\, \mathbf{K}^{-1}\mathbf{k}_*\right) \ . \tag{3.4}$$

[2]Notice that learning cannot be achieved from the likelihood alone: defining a prior over function values is essential to learning.

[3]In Sect. A.2 some algebra useful for deriving (3.2) is given: notice that the likelihood $p(\mathbf{y}|\mathbf{f}, \sigma^2)$ is also Gaussian in $\mathbf{f}$ with mean $\mathbf{y}$.

The predictive distribution of f_* is obtained by integrating out the training function values $\mathbf{f}$ from (3.4) over the posterior distribution (3.2). The predictive distribution is Gaussian:

$$\begin{aligned} p(f_*|\mathbf{y}, \mathbf{x}_*, X, \theta, \sigma^2) &= \int p(f_*|\mathbf{f}, \mathbf{x}_*, X, \theta)\, p(\mathbf{f}|\mathbf{y}, X, \theta, \sigma^2)\, \mathrm{d}\mathbf{f} \\ &\sim \mathcal{N}\left(m(\mathbf{x}_*), v(\mathbf{x}_*)\right) \ , \end{aligned} \tag{3.5}$$

with mean and variance given by:

$$m(\mathbf{x}_*) = \mathbf{k}_*^\top \left(\mathbf{K} + \sigma^2\, \mathbf{I}\right)^{-1} \mathbf{y} \ , \quad v(\mathbf{x}_*) = k_{**} - \mathbf{k}_*^\top \left(\mathbf{K} + \sigma^2\, \mathbf{I}\right)^{-1} \mathbf{k}_* \ . \tag{3.6}$$

Another way of obtaining the predictive distribution of f_* is to augment the evidence with a new element y_* corresponding to the noisy version of f_* and to then write the conditional distribution of y_* given the training targets $\mathbf{y}$. The variance of the predictive distribution of y_* is equal to that of the predictive distribution of f_* (3.6) plus the noise variance σ^2, while the means are identical (the noise has zero mean).

Both if one chooses to learn the hyperparameters or to be Bayesian and do integration, the marginal likelihood of the hyperparameters (or evidence of the observed targets)[4] must be computed. In the first case this quantity will be maximised with respect to the hyperparameters, and in the second case it will be part of the posterior distribution from which the hyperparameters will be sampled. The evidence is obtained by averaging the likelihood over the prior distribution on the function values:

$$p(\mathbf{y}|X, \theta, \sigma^2) = \int p(\mathbf{y}|\mathbf{f})\, p(\mathbf{f}|X, \theta)\, \mathrm{d}\mathbf{f} \sim \mathcal{N}\left(0, \mathbf{K} + \sigma^2\, \mathbf{I}\right) \ . \tag{3.7}$$

Notice that the evidence only differs from the prior over function values in a "ridge" term added to the covariance, that corresponds to the additive Gaussian i.i.d. output noise. Maximum likelihood II learning involves estimating the hyperparameters θ and the noise variance σ^2 by minimising (usually for convenience) the negative log evidence. Let $Q \equiv \left(\mathbf{K} + \sigma^2\, \mathbf{I}\right)$. The cost function and its derivatives are given by:

$$\begin{aligned} \mathcal{L} &= \frac{1}{2} \log|Q| + \frac{1}{2} \mathbf{y}^\top Q^{-1} \mathbf{y} \ , \\ \frac{\partial \mathcal{L}}{\partial \theta_i} &= \frac{1}{2} \operatorname{Tr}\left(Q^{-1} \frac{\partial Q}{\partial \theta_i}\right) - \mathbf{y}^\top Q^{-1} \frac{\partial Q}{\partial \theta_i} Q^{-1} \mathbf{y} \ , \\ \frac{\partial \mathcal{L}}{\partial \sigma^2} &= \frac{1}{2} \operatorname{Tr}\left(Q^{-1}\right) - \mathbf{y}^\top Q^{-1} Q^{-1} \mathbf{y} \ , \end{aligned} \tag{3.8}$$

[4] We will from now on use indistinctly "marginal likelihood" or "evidence" to refer to this distribution.

and one can use some gradient descent algorithm to minimise $\mathcal{L}$ (conjugate gradient gives good results, Rasmussen, 1996).

For Gaussian processes, the computational cost of learning is marked by the need to invert matrix Q and therefore scales with the cube of the number of training cases ($\mathcal{O}(N^3)$). If Q^{-1} is known (obtained from the learning process), the computational cost of making predictions is $\mathcal{O}(n)$ for computing the predictive mean, and $\mathcal{O}(N^2)$ for the predictive variance for each test case. There is a need for approximations that simplify the computational cost if Gaussian Processes are to be used with large training datasets.

3.2 Gaussian Processes as Linear Models

Gaussian Processes correspond to parametric models with an infinite number of parameters. Williams (1997a) showed that infinite neural networks with certain transfer functions and the appropriate priors on the weights are equivalent to Gaussian Processes with a particular "neural network" covariance function. Conversely, any Gaussian Process is equivalent to a parametric model, that can be infinite.

In Sect(s). 3.2.1 and 3.2.2 we establish the equivalence between GPs and linear models. For the common case of GPs with covariance functions that cannot be expressed as a finite expansion, the equivalent linear models are infinite. However, it might still be interesting to approximate such GPs by a finite linear model, which results in *degenerate* Gaussian Processes. In Sect. 3.2.3 we introduce degenerate GPs and explain that they often correspond to inappropriate priors over functions, implying counterintuitive predictive variances. We then show how to modify these degenerate GPs at test time to obtain more appropriate priors over functions.

3.2.1 From Linear Models to GPs

Consider the following extended linear model, where the model outputs are a linear combination of the response of a set of basis functions $\{\phi_j(\mathbf{x}) | j = 1, \ldots, M\} \subset [\mathbb{R}^D \to \mathbb{R}]$:

$$f(\mathbf{x}_i) \;=\; \sum_{j=1}^{M} \phi_j(\mathbf{x}_i)\, w_j = \boldsymbol{\phi}(\mathbf{x}_i)\, \mathbf{w} \;, \qquad\qquad \mathbf{f} \;=\; \Phi\, \mathbf{w} \;, \tag{3.9}$$

where as earlier $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]^\top$ are the function outputs. The weights are organised in a vector $\mathbf{w} = [w_1, \dots, w_M]^\top$, and $\phi_j(\mathbf{x}_i)$ is the response of the j-th basis function to input x_i. $\boldsymbol{\phi}(\mathbf{x}_i) = [\phi_1(\mathbf{x}_i), \dots, \phi_M(\mathbf{x}_i)]$ is a row vector that contains the response of all M basis functions to input $\mathbf{x}_i$ and matrix Φ (sometimes called *design matrix*) has as its i-th row vector $\boldsymbol{\phi}(\mathbf{x}_i)$. Let us define a Gaussian prior over the weights, of the form $p(\mathbf{w}|\mathbf{A}) \sim \mathcal{N}(0, \mathbf{A})$. Since $\mathbf{f}$ is a linear function of $\mathbf{w}$ it has a Gaussian distribution under the prior on $\mathbf{w}$, with mean zero. The prior distribution of $\mathbf{f}$ is:

$$p(\mathbf{f}|\mathbf{A}, \Phi) \sim \mathcal{N}(0, \mathbf{C}) \ , \qquad \mathbf{C} = \Phi \, \mathbf{A} \, \Phi^\top \ . \tag{3.10}$$

The model we have defined corresponds to a Gaussian Process. Now, if the number of basis functions M is smaller than the number of training points N, then $\mathbf{C}$ will not have full rank and the probability distribution of $\mathbf{f}$ will be an elliptical pancake confined to an M-dimensional subspace in the N-dimensional space where $\mathbf{f}$ lives (Mackay, 1997).

Let again $\mathbf{y}$ be the vector of observed training targets, and assume that the output noise is additive Gaussian i.i.d. of mean zero and variance σ^2. The likelihood of the weights is then Gaussian (in $\mathbf{y}$ and in $\mathbf{w}$) given by $p(\mathbf{y}|\mathbf{w}, \Phi, \sigma^2) \sim \mathcal{N}(\Phi \, \mathbf{w}, \sigma^2 \, \mathbf{I})$. The prior over the training targets is then given by

$$p(\mathbf{y}|\mathbf{A}, \Phi, \sigma^2) \sim (0, \sigma^2 \, \mathbf{I} + \mathbf{C}) \ , \tag{3.11}$$

and has a full rank covariance, even if $\mathbf{C}$ is rank deficient.

To make predictions, one option is to build the joint distribution of the training targets and the new test function value and then condition on the targets. The other option is to compute the posterior distribution over the weights from the likelihood and the prior. Williams (1997b) refers to the first option as the "function-space view" and to the second as the "weight-space view". This distinction has inspired us for writing the next two sections.

3.2.1.1 The Parameter Space View.

Using Bayes rule, we find that the posterior is the product of two Gaussians in $\mathbf{w}$, and is therefore a Gaussian distribution:

$$\begin{aligned} &p(\mathbf{w}|\mathbf{y}, \mathbf{A}, \Phi, \sigma^2) = \frac{p(\mathbf{y}|\mathbf{w}, \Phi, \sigma^2) \, p(\mathbf{w}|\mathbf{A})}{p(\mathbf{y}|\mathbf{A}, \Phi, \sigma^2)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \ , \\ &\boldsymbol{\mu} = \sigma^{-2} \, \boldsymbol{\Sigma} \, \Phi^\top \, \mathbf{y} \ , \qquad \boldsymbol{\Sigma} = \left[\sigma^{-2} \, \Phi^\top \Phi + \mathbf{A}^{-1}\right]^{-1} \ . \end{aligned} \tag{3.12}$$

The maximum a posteriori (MAP) estimate of the model weights is given by $\boldsymbol{\mu}$. If we rewrite this quantity as $\boldsymbol{\mu} = [\Phi^\top \Phi + \sigma^2 \, \mathbf{A}]^{-1} \Phi^\top \mathbf{y}$, we can see that the Gaussian assumption on the prior over the weights and on the output noise results in $\boldsymbol{\mu}$ being given by a regularised version of the normal equations. For a new test point $\mathbf{x}_*$, the corresponding function value is $f_* = \boldsymbol{\phi}(\mathbf{x}_*)\,\mathbf{w}$; for making predictions the w's are drawn from the posterior. Since f_* is linear in $\mathbf{w}$, it is quite clear that the predictive distribution $p(f_*|\mathbf{y}, \mathbf{A}, \Phi, \sigma^2)$ is Gaussian, with mean and variance given by:

$$m(\mathbf{x}_*) = \boldsymbol{\phi}(\mathbf{x}_*)^\top \boldsymbol{\mu} \ , \qquad v(\mathbf{x}_*) = \boldsymbol{\phi}(\mathbf{x}_*)^\top \boldsymbol{\Sigma}\, \boldsymbol{\phi}(\mathbf{x}_*) \ . \tag{3.13}$$

We can rewrite the posterior covariance using the matrix inversion lemma (see App. A.1) as $\boldsymbol{\Sigma} = \mathbf{A} - \mathbf{A}[\sigma^2\,\mathbf{I} + \Phi\,\mathbf{A}\,\Phi^\top]^{-1}\,\mathbf{A}$. This expression allows us to rewrite the predictive mean and variance as:

$$\begin{aligned} m(\mathbf{x}_*) &= \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}\,\Phi^\top [\sigma^2\,\mathbf{I} + \Phi\,\mathbf{A}\,\Phi^\top]^{-1}\mathbf{y} \ , \\ v(\mathbf{x}_*) &= \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}\,\boldsymbol{\phi}(\mathbf{x}_*) - \boldsymbol{\phi}(\mathbf{x}_*)^\top \mathbf{A}\,\Phi^\top [\sigma^2\,\mathbf{I} + \Phi\,\mathbf{A}\,\Phi^\top]^{-1}\Phi\,\mathbf{A}\,\boldsymbol{\phi}(\mathbf{x}_*) \ , \end{aligned} \tag{3.14}$$

which will be useful for relating the parameter space view to the GP view.

3.2.1.2 The Gaussian Process View.

There exists a Gaussian Process that is equivalent to our linear model with Gaussian priors on the weights given by (3.9). The covariance function of the equivalent GP is given by:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^\top \mathbf{A}\,\boldsymbol{\phi}(\mathbf{x}_j) = \sum_{k=1}^{M}\sum_{l=1}^{M} \mathbf{A}_{kl}\,\phi_k(\mathbf{x}_i)\,\phi_l(\mathbf{x}_j) \ . \tag{3.15}$$

The covariance matrix of the prior over training function values is given by $\mathbf{K} = \Phi\,\mathbf{A}\,\Phi^\top$ and we recover the same prior as in (3.10). Taking the same noise model as previously, the prior over targets is identical to (3.11).

Given a new test input $\mathbf{x}_*$, the vector of covariances between $\mathbf{f}_*$ and the training function values is given by $\mathbf{k}_* = \Phi\,\mathbf{A}\,\boldsymbol{\phi}(\mathbf{x}_*)$ and the prior variance of f_* is $k_{**} = \boldsymbol{\phi}(\mathbf{x}_*)\,\mathbf{A}\,\boldsymbol{\phi}(\mathbf{x}_*)$. Plugging these expressions into the equations for the predictive mean and variance of a GP (3.6) one recovers the expressions given by (3.14) and (3.13). The predictive mean and variance of a GP with covariance function given by (3.15) are therefore identical to the predictive mean and variance of the linear model.

A fundamental property of the GP view of a linear model is that the set of M basis functions appear *exclusively* as inner products. Linear models where M is infinite are thus tractable under the GP view, provided that the basis functions and the prior over the weights are appropriately chosen. By appropriately chosen we mean such that a generalised dot product exists in feature space, that allows for the use of the *"kernel trick"*. Schölkopf and Smola (2002) provide with extensive background on kernels and the *"kernel trick"*.

Let us reproduce here an example given by Mackay (1997). Consider a one-dimensional input space, and let us use squared exponential basis functions $\phi_c(x_i) = \exp(-\frac{1}{2}(x_i - c)^2/\lambda^2)$, where c is a given centre in input space and λ is a known lengthscale. Let us also define an isotropic prior over the weights, of the form $\mathbf{A} = \sigma_w^2\,\mathbf{I}$. We want to make M go to infinity, and assume for simplicity uniformly spaced basis functions. To make sure that the integral converges, we set variance of the prior over the weights to $\sigma_w^2 = s/\Delta M$, where ΔM is the density of basis functions in the input space. The covariance function is given by:

$$
\begin{aligned}
k(x_i, x_j) &= s \int_{c_{min}}^{c_{max}} \phi_c(x_i)\,\phi_c(x_j)\,\mathrm{d}c \ , \\
&= s \int_{c_{min}}^{c_{max}} \exp\left[-\frac{(x_i - c)^2}{2\lambda^2}\right] \exp\left[-\frac{(x_j - c)^2}{2\lambda^2}\right] \mathrm{d}c \ .
\end{aligned}
\tag{3.16}
$$

Letting the limits of the integral go to infinity, we obtain the integral of the product of two Gaussians (but for a normalisation factor), and we can use the algebra from Sect. A.2 to obtain:

$$
k(x_i, x_j) = s\,\sqrt{\pi\lambda^2}\,\exp\left[-\frac{(x_i - x_j)^2}{4\lambda^2}\right] \ , \tag{3.17}
$$

which is the squared exponential covariance function that we presented in (3.1). We now see that a GP with this particular covariance function is equivalent to a linear model with infinitely many squared exponential basis functions.

In the following we will show that for any valid covariance function, a GP has an equivalent linear model. The equivalent linear model will have infinitely many weights if the GP has a covariance function that has no finite expansion.

3.2.2 From GPs to Linear Models

We have just seen how to go from any linear model, finite or infinite, to an equivalent GP. We will now see how to go the opposite way, from an arbitrary GP to an equivalent linear model, which will in general be infinite and will be finite only for particular choices of the covariance function.

We start by building a linear model where all the function values considered (training *and* test inputs) are equal to a linear combination of the rows of the corresponding covariance matrix of the GP we wish to approximate, computed with the corresponding covariance function $K(\mathbf{x}_i, \mathbf{x}_j)$. As in Sect. 3.1, the covariance function is parameterised by the hyperparameters θ. A Gaussian prior distribution is defined on the model weights, with zero mean and covariance equal to the inverse of the covariance matrix:

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{w} \\ w_* \end{bmatrix} \quad , \quad p\left(\begin{bmatrix} \mathbf{w} \\ w_* \end{bmatrix} \middle| \mathbf{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**} \end{bmatrix}^{-1}\right) \quad . \tag{3.18}$$

To compute the corresponding prior over function values we need to integrate out the weights $[\mathbf{w}, w_*]^\top$ from the left expression in (3.18) by averaging over the prior (right expression in (3.18)):

$$\begin{aligned} &p\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \middle| \mathbf{x}_*, X, \theta\right) \\ &= \int \delta\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} - \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{w} \\ w_* \end{bmatrix}\right) p\left(\begin{bmatrix} \mathbf{w} \\ w_* \end{bmatrix} \middle| \mathbf{x}_*, X, \theta\right) \mathrm{d}\mathbf{w} \\ &\sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**} \end{bmatrix}\right) \quad , \end{aligned} \tag{3.19}$$

and we recover exactly the same prior over function values as for the Gaussian Process, see (3.3).

Notice that for the linear model to correspond to the full GP two requirements need to be fulfilled:

1. There must be a weight associated to each training input.
2. There must be a weight associated to each possible test input.

Since there are as many weights as input instances, we consider that there is an infinite number of weights of which we only use as many as needed and qualify such a linear model of *infinite*.

Of course, for covariance functions that have a finite expansion in terms of M basis functions, the rank of the covariance matrix will never be greater than M and the equivalent linear model can be readily seen to be finite, with M basis functions. A trivial example is the case where the covariance function is built from a finite linear model with Gaussian priors on the weights. The linear model equivalent to a GP is only infinite if the covariance function of the GP has no finite expansion. In that case, independently of the number of training and test cases considered, the covariance matrix of the prior (independently of its size) will always have full rank.[5]

It becomes evident how one should deal with GPs that have an equivalent finite linear model. If there are more training cases than basis functions, $N > M$, then the finite linear model should be used. In the case where there are less training cases than basis functions, $M > N$, it is computationally more interesting to use the GP.

One strong motivation for the use of Gaussian Processes is the freedom to directly specify the covariance function. In practice, common choices of GP priors imply covariance functions that do not have a finite expansion. For large datasets, this motivates the approximation of the equivalent infinite linear model by a finite or sparse one. The approximated GP is called Reduced Rank GP since its covariance matrix has a maximum rank equal to the number of weights in the finite linear model.

We will see later in Sect. 3.3 that the finite linear approximation is built by relaxing the requirement of a weight being associated to each training input, resulting in training inputs with no associated weight. This relaxation should only be done at training time. In the next section we show the importance of maintaining the requirement of having a weight associated to each test input.

3.2.3 "Can I Skip w_*?" or Degenerate Gaussian Processes

One may think that having just "as many weights as training cases" with no additional weight w_* associated to each test case gives the same prior as a full GP. It does only for the function evaluated at the training inputs, but it does not anymore for any additional function value considered. Indeed, if we posed $\mathbf{f} = \mathbf{K}\,\mathbf{w}$ with a prior over the weights given by $p(\mathbf{w}|X,\theta) \sim \mathcal{N}(0, \mathbf{K}^{-1})$, we would obtain that the corresponding prior over the training function values is $p(\mathbf{f}|X,\theta,\sigma^2) \sim \mathcal{N}(0,\mathbf{K})$. It is true that the linear model would be equivalent to the GP, but only when the function values considered are in $\mathbf{f}$. Without addition of w_*, the linear model and prior over function values are respectively given by:

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} = \begin{bmatrix} \mathbf{K} \\ \mathbf{k}_*^\top \end{bmatrix} \cdot \mathbf{w} \ , \qquad p\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \middle| \mathbf{x}_*, X, \theta\right) \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k}_* \\ \mathbf{k}_*^\top & \mathbf{k}_*^\top \mathbf{K}^{-1} \mathbf{k}_* \end{bmatrix}\right) \ . \tag{3.20}$$

[5] The covariance matrix can always be made rank deficient by replicating a function value in the joint prior, but we do not see any reason to do this in practice.

The prior over the new function values f_* differs now from that of the full GP. Notice that the *prior* variance of f_* *depends* on the training inputs: for the common choice of an RBF-type covariance function, if $\mathbf{x}_*$ is far from the training inputs, then there is *a priori* no signal, that is f_* is zero without uncertainty! Furthermore, the distribution of $\mathbf{f}_*$ conditioned on the training function outputs, which for the full GP is given by (3.4), has now become:

$$p(f_*|\mathbf{f}, \mathbf{x}_*, X, \theta) \sim \mathcal{N}\left(\mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{f}, 0\right) \quad . \tag{3.21}$$

Given $\mathbf{f}$, any additional function value f_* is not a random variable anymore, since its conditional distribution has zero variance: f_* is fully determined by $\mathbf{f}$.

If $\mathbf{w}$ has a fixed finite size, the prior over functions implied by the linear model ceases to correspond to the GP prior. The joint prior over sets of function values is still Gaussian, which raises the question "is this still a GP?". We choose to call such a *degenerate* process a "degenerate Gaussian Process".

In the case of localised (decaying) covariance functions, such as the squared exponential, degenerate GPs produce a predictive distribution that has maximal variability around the training inputs, while the predictive variance fades to the noise level as one moves away from them. We illustrate this effect on Fig. 3.1. We plot the predictive standard deviation of a full GP and its degenerate counterpart for various test points. The training set consists of 5 points: both models have thus 5 weights associated to the training set. The full GP has an additional weight, associated to each test point one at a time. Though it might be a reasonable prior in particular contexts, we believe that it is in general *inappropriate* to have smaller predictive variance far away from the observed data. We believe that appropriate priors are those under which the predictive variance is reduced when the test inputs approach training inputs. In the case of non-localised covariance functions, such as those for linear models or neural networks, the predictive uncertainty does not decay as one moves away from the training inputs, but rather increases. However, it is in that case much harder to analyse the goodness of the prior over function evaluations. Would it be reasonable to assume that the predictive uncertainty increases monotonically as one moves away from the training inputs? Or is our assumption that it should saturate more appropriate?

3.3 Finite Linear Approximations

As we have discussed in Sect. 3.2.1, a weight must be associated to each test case to avoid inappropriate priors that produce inappropriate predictive errorbars. However, the requirement of each training case having a weight associated to it

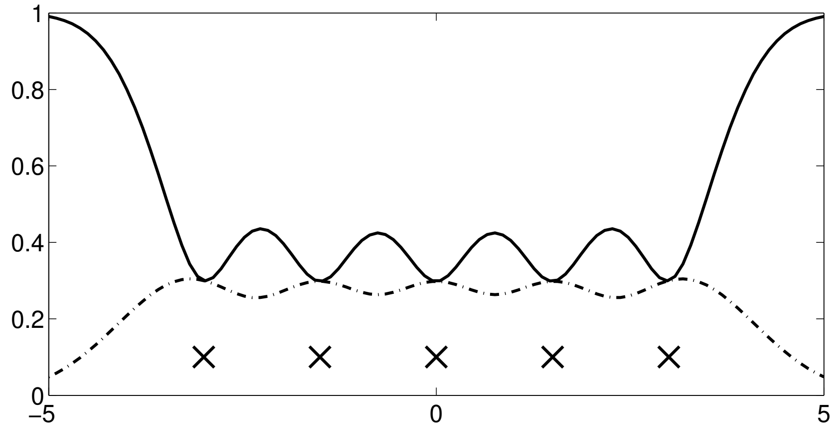

Figure 3.1: Predictive standard deviation for a full GP (solid line) and for a degenerate GP (slash-dotted line). The hyperparameters θ_i are all set to 1. The crosses indicate the horizontal location of the 5 training inputs.

can be relaxed. For computational reasons it might be interesting to approximate, *at training time*, a GP by a finite linear model with less weights than training cases. The model and the prior on the weights are respectively given by:

$$\mathbf{f} = \mathbf{K}_{NM}\,\mathbf{w}_M \ , \qquad\qquad p(\mathbf{w}_M|X,\theta) \sim \mathcal{N}(0, \mathbf{K}_{MM}^{-1}) \ , \tag{3.22}$$

The subscripts M and N are used to indicate the dimensions: w_M is of size $M \times 1$ and $\mathbf{K}_{NM}$ of size $N \times M$; in the following we will omit these subscripts where unnecessary or cumbersome. Sparseness arises when $M < N$: the induced prior over training function values is $p(\mathbf{f}|X,\theta) \sim \mathcal{N}\left(0, \mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\,\mathbf{K}_{NM}^{\top}\right)$, and the rank of the covariance matrix is at most M. We call such an approximation a Reduced Rank Gaussian Process (RRGP).

The M inputs associated to the weights in $\mathbf{w}_M$ do not need to correspond to training inputs. They can indeed be any set of arbitrary points in input space. We will call such points *support inputs* (in recognition to the large amount of work on sparse models done by the Support Vector Machines community). In this paper we will adopt the common restriction of selecting the support set from the training inputs. We discuss ways of selecting the support points in Sect. 3.3.4.

Learning an RRGP consists both in learning the hyperparameters of the covariance function and in selecting the support set. In practice however, it is hard to do both simultaneously. Besides the technical difficulties of the optimisation process (observed for example by Csató (2002)), there is the fundamental issue of having an excessive amount of flexibility that may lead to over-fitting (observed for example by Rasmussen (2002) and Seeger et al. (2003)). Smola and Bartlett (2001) address the issue of selecting the support set (Sect. 3.3.5), assuming that the covariance hyperparameters are given. However, we show that they do this in a way that does not guarantee generalisation and we propose an alternative theoretically more sound approach in Sect. 3.3.4. In the next section we show how to learn the hyperparameters of the covariance function for the RRGP for a fixed support set. We also show how to make predictions under a degenerate GP, that is, without an additional weight for the new test case, and with the inclusion of a new weight that ensures appropriate predictive variances.

3.3.1 Learning a Reduced Rank Gaussian Process

The likelihood of the weights is Gaussian in $\mathbf{y}$ and is a linear combination of w_M, given by $p(\mathbf{y}|X, \theta, \mathbf{w}_M, \sigma^2) \sim \mathcal{N}(\mathbf{K}_{NM}\,\mathbf{w}_M, \sigma^2\,\mathbf{I})$, where σ^2 is again the white noise variance. The marginal likelihood of the hyperparameters of the full GP is given by (3.7). For the sparse finite linear approximation, the marginal likelihood is obtained by averaging the weights out of the likelihood over their prior:

$$\begin{aligned} p(\mathbf{y}|X,\theta,\sigma^2) &= \int p(\mathbf{y}|X,\theta,\mathbf{w}_M,\sigma^2)\, p(\mathbf{w}_M|X,\theta)\,\mathrm{d}\mathbf{w}_M \\ &\sim \mathcal{N}\left(0, \sigma^2\,\mathbf{I} + \mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\mathbf{K}_{NM}^\top\right) \quad . \end{aligned} \tag{3.23}$$

As expected, for the case where the support set comprises all training inputs and $M = N$, we recover the full Gaussian Process.

Let us define $\tilde{Q} \equiv \left[\sigma^2\,\mathbf{I} + \mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\mathbf{K}_{NM}^\top\right]$, the covariance of the RRGP evidence. Maximum likelihood learning of the hyperparameters can be achieved by minimising the negative log evidence. The cost function and its derivatives are given by (3.8) where Q is replaced by $\tilde{Q}$. Since the simple linear algebra involved can be tedious, we give here the explicit expression of the different terms. For the terms involving $\log|\tilde{Q}|$ we have:

$$\begin{aligned} \log|\tilde{Q}| &= (N-M)\,log(\sigma^2) + \log\left|\mathbf{K}_{NM}^\top\,\mathbf{K}_{NM} + \sigma^2\,\mathbf{K}_{MM}\right| \quad , \\ \frac{\partial \log|\tilde{Q}|}{\partial\theta_i} &= \mathrm{Tr}\left[\tilde{Q}^{-1}\,\frac{\partial\tilde{Q}}{\partial\theta_i}\right] = 2\,\mathrm{Tr}\left[\frac{\partial\mathbf{K}_{NM}}{\partial\theta_i}\,Z^\top\right] - \mathrm{Tr}\left[\mathbf{K}_{MM}^{-1}\,\mathbf{K}_{NM}^\top\,Z\,\frac{\partial\mathbf{K}_{NM}}{\partial\theta_i}\right] \quad , \\ \frac{\partial \log|\tilde{Q}|}{\partial\sigma^2} &= \frac{n-M}{\sigma^2} + \mathrm{Tr}\left[Z_{MM}\right] \quad , \end{aligned}$$

(3.24)

where we have introduced $Z \equiv \mathbf{K}_{NM} \left[\mathbf{K}_{NM}^\top \mathbf{K}_{NM} + \sigma^2 \mathbf{K}_{MM}\right]^{-1}$. For the terms involving $\tilde{Q}^{-1}$ we have:

$$\begin{aligned}
\mathbf{y}^\top \tilde{Q}^{-1} \mathbf{y} &= \left(\mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top Z \mathbf{K}_{NM}^\top \mathbf{y}\right) / \sigma^2 \ , \\
\frac{\partial \mathbf{y}^\top \tilde{Q}^{-1} \mathbf{y}}{\partial \theta_i} &= \mathbf{y}^\top Z \frac{\partial \mathbf{K}_{MM}}{\partial \theta_i} Z^\top \mathbf{y} - 2\, \mathbf{y}^\top \left(\mathbf{I} - Z \mathbf{K}_{NM}^\top\right) \frac{\partial \mathbf{K}_{NM}}{\partial \theta_i} Z^\top \mathbf{y}/\sigma^2 \ , \\
\frac{\partial \mathbf{y}^\top \tilde{Q}^{-1} \mathbf{y}}{\partial \sigma^2} &= -\mathbf{y}^\top \mathbf{y}/\sigma^4 + \mathbf{y}^\top Z \mathbf{K}_{NM}^\top \mathbf{y}/\sigma^4 + \mathbf{y}^\top Z \mathbf{K}_{MM} Z^\top \mathbf{y}/\sigma^2 \ .
\end{aligned} \tag{3.25}$$

The hyperparameters and the output noise variance can be learnt by using the expressions we have given for the negative log marginal likelihood and its derivatives in conjunction with some gradient descent algorithm. The computational complexity of evaluating the evidence and its derivatives is $\mathcal{O}(NM^2 + NDM)$, which is to be compared with the corresponding cost of $\mathcal{O}(N^3)$ for the full GP model.

3.3.2 Making Predictions without w_*

The posterior over the weights associated to the training function values is $p(\mathbf{w}_M | \mathbf{y}, \mathbf{K}_{NM}, \sigma^2) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with:

$$\boldsymbol{\mu} = \sigma^{-2} \boldsymbol{\Sigma}\, \mathbf{K}_{NM}^\top \mathbf{y} \ , \qquad \boldsymbol{\Sigma} = \left[\sigma^{-2} \mathbf{K}_{NM}^\top \mathbf{K}_{NM} + \mathbf{K}_{MM}\right]^{-1} \ . \tag{3.26}$$

At this point one can choose to make predictions right now, based on the posterior of $\mathbf{w}_M$ and without adding an additional weight w_* associated to the new test point $\mathbf{x}_*$. As discussed in Sect. 3.2.3, this would correspond to a degenerate GP, leading to inappropriate predictive variance. The predictive mean on the other hand can still be a reasonable approximation to that of the GP: Smola and Bartlett (2001) approximate the predictive mean exactly in this way. The expressions for the predictive mean and variance, when not including w_*, are respectively given by:

$$m(\mathbf{x}_*) = \mathbf{k}(\mathbf{x}_*)^\top \boldsymbol{\mu}, \qquad v(\mathbf{x}_*) = \sigma^2 + \mathbf{k}(\mathbf{x}_*)^\top \boldsymbol{\Sigma}\, \mathbf{k}(\mathbf{x}_*). \tag{3.27}$$

$\mathbf{k}(\mathbf{x}_*)$ denotes the $m \times 1$ vector $[K(\mathbf{x}_*, \mathbf{x}_1), \ldots, K(\mathbf{x}_*, \mathbf{x}_M)]^\top$ of covariances between $\mathbf{x}_*$ and at the M support inputs (as opposed to $\mathbf{k}_*$ which is the $N \times 1$ vector of covariances between $\mathbf{x}_*$ and at the N training inputs). Note that if no sparseness is enforced, $(M = N)$, then $\boldsymbol{\mu} = (\mathbf{K}_{NN} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$ and the predictive mean $m(\mathbf{x}_*)$ becomes identical to that of the full GP. Also, note that for decaying covariance functions,[6] if $\mathbf{x}_*$ is far away from the selected training inputs, the predictive variance collapses to the output noise level, which we have defined as an inappropriate prior.

The computational cost of predicting without w_* is an initial $\mathcal{O}(NM^2)$ to compute $\mathbf{\Sigma}$, and then an additional $\mathcal{O}(M)$ for the predictive mean and $\mathcal{O}(M^2)$ for the predictive variance per test case.

3.3.3 Making Predictions with w_*

To obtain a better approximation to the full GP, especially in terms of the predictive variance, we add an extra weight w_* to the model for each test input $\mathbf{x}_*$. Unless we are interested in the predictive covariance for a set of test inputs, it is enough to add one single w_* at a time. The total number of weights is therefore only augmented by one for any test case.

For a new test point, the mean and covariance matrix of the new posterior over the augmented weights vector are given by:

$$\begin{aligned}
\boldsymbol{\mu}_* &= \sigma^{-2}\mathbf{\Sigma}_* \begin{bmatrix} \mathbf{K}_{NM}^\top \\ \mathbf{k}_*^\top \end{bmatrix} \mathbf{y} \ , \\
\mathbf{\Sigma}_* &= \begin{bmatrix} \mathbf{\Sigma}^{-1} & \mathbf{k}(\mathbf{x}_*) + \sigma^{-2}\,\mathbf{K}_{NM}^\top \mathbf{k}_* \\ \mathbf{k}(\mathbf{x}_*)^\top + \sigma^{-2}\,\mathbf{k}_*^\top \mathbf{K}_{NM} & k_{**} + \sigma^{-2}\,\mathbf{k}_*^\top \mathbf{k}_* \end{bmatrix}^{-1} \ .
\end{aligned} \tag{3.28}$$

and the computational cost of updating the posterior and computing the predictive mean and variance is $\mathcal{O}(NM)$ for each test point. The most expensive operation is computing $\mathbf{K}_{NM}^\top \mathbf{k}_*$ with $\mathcal{O}(NM)$ operations. Once this is done and given that we have previously computed $\mathbf{\Sigma}$, computing $\mathbf{\Sigma}_*$ can be efficiently done using inversion by partitioning in $\mathcal{O}(M^2)$ (see Sect. A.1 for the details). The predictive mean and variance can be computed by plugging the updated posterior parameters (3.28) into (3.27), or alternatively by building the updated joint prior over the training and new test function values. We describe in detail the algebra involved in the second option in App. A.5. The predictive mean and variance when including w_* are respectively given by:

$$\begin{aligned}
m_*(\mathbf{x}_*) &= \mathbf{k}_*^\top \left[\mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\,\mathbf{K}_{NM}^\top + \sigma^2\,\mathbf{I} + \mathbf{v}_*\mathbf{v}_*^\top/c_*\right]^{-1} \mathbf{y} \ , \\
v_*(\mathbf{x}_*) &= \sigma^2 + k_{**} + \mathbf{k}_*^\top \left[\mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\,\mathbf{K}_{NM}^\top + \sigma^2\,\mathbf{I} + \mathbf{v}_*\mathbf{v}_*^\top/c_*\right]^{-1} \mathbf{k}_* \ .
\end{aligned} \tag{3.29}$$

[6] Covariance functions whose value decays with the distance between the two arguments. One example is the squared exponential covariance function described in Sect. 3.1. Decaying covariance functions are very commonly encountered in practice.

where $\mathbf{v}_* \equiv \mathbf{k}_* - \mathbf{K}_{NM}\, \mathbf{K}_{MM}^{-1}\, \mathbf{k}(\mathbf{x}_*)$ is the difference between the actual and the approximated covariance of f_* and $\mathbf{f}$, and $c_* \equiv k_{**} - \mathbf{k}(\mathbf{x}_*)^\top \mathbf{K}_{MM}^{-1}\, \mathbf{k}(\mathbf{x}_*)$ is the predictive variance at $\mathbf{x}_*$ of a full GP with the support inputs as training inputs.

3.3.4 Selecting the Support Points

One way of addressing the problem of selecting the M support inputs is to select them from among the N training inputs. The number of possible sets of support inputs is combinatorial, C_N^M.[7] Since we will typically be interested in support sets much smaller than the training sets ($M < N$), this implies that the number of possible support sets is roughly exponential in M. Ideally one would like to evaluate the evidence for the finite linear model approximation (3.23), for each possible support input set, and then select the set that yields a higher evidence. In most cases however, this is impractical due to computational limitations. One suboptimal solution is to opt for a greedy method: starting with an empty subset, one includes the input that results in a maximal increase in evidence. The greedy method exploits the fact that the evidence can be computed efficiently when a case is added (or deleted) to the support set.

Suppose that a candidate input $\mathbf{x}_i$ from the training set is considered for inclusion in the support set. The new marginal likelihood is given by:

$$\mathcal{L}_i = \frac{1}{2}\log|\tilde{Q}_i| + \frac{1}{2}\mathbf{y}^\top \tilde{Q}_i^{-1}\,\mathbf{y}\ , \qquad \tilde{Q}_i \equiv \sigma^2\,\mathbf{I} + \mathbf{K}_{N\tilde{M}}\,\mathbf{K}_{\tilde{M}\tilde{M}}^{-1}\,\mathbf{K}_{N\tilde{M}}^\top\ , \quad (3.30)$$

where $\tilde{M}$ is the set of $M+1$ elements containing the M elements in the current support set plus the new case $\mathbf{x}_i$. $\tilde{Q}_i$ is the updated covariance of the evidence of the RRGP augmented with $\mathbf{x}_i$. Let us deal separately with the two terms in the evidence. The matrix inversion lemma allows us to rewrite $\tilde{Q}_i$ as:

$$\tilde{Q}_i = \sigma^{-2}\,\mathbf{I} - \sigma^{-4}\,\mathbf{K}_{N\tilde{M}}\,\mathbf{\Sigma}_i\,\mathbf{K}_{N\tilde{M}}^\top\ , \qquad \mathbf{\Sigma}_i = \left[\mathbf{K}_{N\tilde{M}}^\top \mathbf{K}_{N\tilde{M}}/\sigma^2 + \mathbf{K}_{\tilde{M}\tilde{M}}\right]^{-1}\ , \quad (3.31)$$

where $\mathbf{\Sigma}_i$ is the covariance of the posterior over the weights augmented in w_i, the weight associated to $\mathbf{x}_i$. Notice that $\mathbf{\Sigma}_i$ is the same expression as $\mathbf{\Sigma}_*$ in (3.28) if one replaces the index $*$ by i. In both cases we augment the posterior in the same way. Computing $\mathbf{\Sigma}_i$ from $\mathbf{\Sigma}$ costs therefore only $\mathcal{O}(NM)$.

The term of $\mathcal{L}$ quadratic in $\mathbf{y}$ can be rewritten as:

$$\mathcal{Q}_i = \frac{1}{2\sigma^2}\,\mathbf{y}^\top\mathbf{y} - \frac{1}{2\sigma^4}\,\mathbf{y}^\top\mathbf{K}_{N\tilde{M}}\,\mathbf{\Sigma}_i\,\mathbf{K}_{N\tilde{M}}^\top\,\mathbf{y}\ , \quad (3.32)$$

[7] C_N^M is "N choose M": the number of combinations of M elements out of N without replacement and where the order does not matter.

and can be computed efficiently in $\mathcal{O}(NM)$ if $\mathbf{\Sigma}$ and $\mathbf{K}_{NM}^\top \mathbf{y}$ are known. In Sect. A.3 we provide the expressions necessary for computing $\mathcal{Q}_i$ incrementally in a robust manner from the Cholesky decomposition of $\mathbf{\Sigma}$. In Sect. 3.3.5 we describe Smola and Bartlett's Sparse Greedy Gaussian Process (SGGP) Regression which uses $\mathcal{Q}_i$ solely as objective function for selecting the support set in a greedy manner.

The term of $\mathcal{L}$ that depends on $\log|\tilde{Q}_i|$ can be expressed as:

$$\mathcal{G}_i = \frac{1}{2}\left[\log|\mathbf{\Sigma}_i| - \log|\mathbf{K}_{\tilde{M}\tilde{M}}| + N\log\sigma^2\right] \quad , \tag{3.33}$$

and computed at a cost of $\mathcal{O}(NM)$ (the cost of computing $\mathbf{K}_{NM}^\top \mathbf{k}_i$). The algebra in Sect. A.3 can be used to update the determinants from the incremental Cholesky decompositions at no additional cost.

The overall cost of evaluating the evidence for each candidate point for the support set is $\mathcal{O}(NM)$. In practice, we may not want to explore the whole training set in search for the best candidate, since this would be too costly. We may restrict ourselves to exploring some reduced random subset.

3.3.5 Sparse Greedy Gaussian Process Regression

Smola and Bartlett (2001) and Schölkopf and Smola (2002) present a method to speed up the prediction stage for Gaussian processes. They propose a sparse greedy techniques to approximate the Maximum a Posteriori (MAP) predictions, treating separately the approximation of the predictive mean and that of the predictive variance.

For the predictive mean, Smola and Bartlett adopt a finite linear approximation of the form given by (3.22), where no extra weight w_* associated to the test input is added. Since this is a degenerate GP, it is understandable that they only use it for approximating the predictive mean: we now know that the predictive uncertainties of degenerate GPs are inappropriate.

The main contribution of their paper is to propose a method for selecting the M inputs in the support set from the N training inputs. Starting from a full posterior distribution (as many weights as training inputs), they aim at finding a sparse weight vector (with only M non-zero entries) with the requirement that *the posterior probability at the approximate solution be close to the maximum of the posterior probability* (quoted from (Schölkopf and Smola, 2002, Sect. 16.4.3)). Since the optimal strategy has again a prohibitive cost, they propose a greedy method where the objective function is the full posterior evaluated at the optimal weights vector with only M non-zeros weighs, those corresponding to the inputs in the support set.

The posterior on $\mathbf{w}_N$ (full posterior) is given by (3.26), where $M = N$, i.e. matrix $\mathbf{K}_{NM}$ is replaced by the full $N \times N$ matrix $\mathbf{K}$. The objective function used in (Smola and Bartlett, 2001; Schölkopf and Smola, 2002) is the part of the negative log posterior that depends on $\mathbf{w}_N$, which is the following quadratic form:

$$-\mathbf{y}^\top \mathbf{K}_{NM}\, \mathbf{w}_M + \frac{1}{2}\, \mathbf{w}_M^\top \left[\mathbf{K}_{NM}^\top \mathbf{K}_{NM} + \sigma^2\, \mathbf{K}_{MM}\right] \mathbf{w}_M \ , \tag{3.34}$$

where as usual $\mathbf{w}_M$ denotes the part of $\mathbf{w}_N$ that hasn't been clamped to zero. Notice that it is essential for the objective function to be the full posterior evaluated at a sparse $\mathbf{w}_N$, rather than the posterior on $\mathbf{w}_M$ (given by (3.26) with indeed $M \neq N$). In the latter case, only the log determinant of the covariance would play a rôle in the posterior, since $\mathbf{w}_M$ would have been made equal to the posterior mean, and we would have a completely different objective function from that in (Smola and Bartlett, 2001; Schölkopf and Smola, 2002).

Given two candidates to the support set, the one resulting in a support set for which the minimum of (3.34) is smaller is chosen. The minimum of (3.34) is given by:

$$-\frac{1}{2}\mathbf{y}^\top\, \mathbf{K}_{NM}\, \left[\mathbf{K}_{NM}^\top \mathbf{K}_{NM} + \sigma^2\, \mathbf{K}_{MM}\right]^{-1} \mathbf{K}_{NM}^\top\, \mathbf{y} \ , \tag{3.35}$$

and it is in fact this quantity that is minimised with respect to the M elements in the support set in a greedy manner. The expression given in (3.35) with $M \neq N$ is in fact an upper bound to the same expression with $M = N$, which corresponds to selecting the whole training set as active set. Smola and Bartlett (2001); Schölkopf and Smola (2002) also provide a lower bound to the latter, which allows them to give a stop criterion to the greedy method based on the relative difference between upper and lower bound. The computational cost of evaluating the expression given in (3.35) for each candidate to the support set is $\mathcal{O}(NM)$, and use can be made of an incremental Cholesky factorisation for numerical stability. The expressions in Sect. A.3 can be used. The computational cost is therefore the same for the SGGP method as for the greedy approach based on maximising the evidence that we propose in Sect. 3.3.4.

3.3.5.1 Why Does it Work?

One might at this point make abstraction from the algorithmic details, and ask oneself the fair question of why obtaining a sparse weight vector that evaluated under the posterior over the full weight vector yields a probability close to that of the non-sparse solution is a good approximation. Along the same lines, one may wonder whether the stopping criterion proposed relates in any way with good generalisation.

It turns out that the method often works well in practice, in a very similar way as our proposed greedy criterion based on maximising the evidence. One explanation for the SGGP method to select meaningful active sets is that it is in fact minimising a part of the negative log evidence $\mathcal{L}_i$, given by (3.30). Indeed, notice that minimising the objective function given by (3.35) is exactly equivalent to minimising the part of the negative log evidence quadratic in $\mathbf{y}$ given by (3.32). So why would the method work if it only maximises $\mathcal{Q}_i$ (3.32), the part of $\mathcal{L}_i$ that has to do with fitting the data, and ignores $\mathcal{G}_i$ (3.33), the part that enforces regularisation? We believe that over-fitting will seldom happen because M is typically significantly smaller than N, and that therefore we are selecting from a family of models that are all very simple. In other words, it is the sparsity itself that guarantees some amount of regularisation, and therefore $\mathcal{G}_i$ can be often safely omitted from the negative log evidence. However, as we will see in what follows, the SGGP can fail and indeed over-fit. The problem is that the SGGP fails to provide a valid stopping criterion for the process of adding elements to the support set.

3.3.5.2 But, How Much Sparsity?

If sparsity seemingly ensures generalisation, then it would also seem that a criterion is needed to know *the minimum sparsity level required*. In other words, we need to know how many inputs it is safe to include in the support set. (Smola and Bartlett, 2001; Schölkopf and Smola, 2002) use a measure they call the "gap", which is the relative difference between the upper and lower bound on the negative log posterior. They choose an arbitrary threshold below which they consider that the approximate posterior has been maximised to a value close enough to the maximum of the full posterior. Once again we fail to see what such a criterion has to do with ensuring generalisation, and we are not the only ones: Schwaighofer and Tresp (2003) report "we did not observe any correlation between the gap and the generalisation performance in our experiments". It might be that for well chosen hyperparameters of the covariance, or for datasets that do not lend themselves to sparse approximations, keeping on adding cases to the support set cannot be harmful. Yet the SGGP does not allow learning the hyperparameters, and those must be somehow guessed (at least not in a direct way).

We provide a simple toy example (Fig. 3.2) in which the value of minimising the negative log evidence becomes apparent. We generate 100 one-dimensional training inputs, equally spaced from -10 to 10. We generate the corresponding training inputs by applying the function $\sin(x)/x$ to the inputs, and adding Gaussian noise of variance 0.01. We generate the test data from 1000 test inputs equally spaced between -12 and 12. We use a squared exponential covariance function as given by (3.1), and we set the hyperparameters in the following way: the lengthscale is $\theta_1 = 1$, the prior standard deviation of the output signal is $\theta_2 = 1$ and the noise variance is $\sigma^2 = \theta_3 = 0.01$. Note that we provide the model with the *actual* variance of the noise. We apply the greedy strategy for selecting the support set by minimising in one case the negative log evidence and in the other case the negative log posterior. Interesting things happen. We plot the test squared error as a function of M, the size of the support set for both greedy strategies. Both have a minimum for support sets of size around 8 to 10 elements, and increase again as for larger support sets. Additionally, we compute the negative log evidence as a function of M, and we see that it has a minimum around the region where the test error is minimal. This means that we can actually use the evidence to determine good levels of sparsity. We also plot the "gap" as a function of M, and indicate the location of the arbitrary threshold of 0.025 used by Smola and Bartlett (2001); Schölkopf and Smola (2002). The gap cannot provide us with useful information in any case, since it is *always* a monotonically decreasing function of M! The threshold is absolutely arbitrary, and has no relation to the expected generalisation of the model.

3.3.5.3 Approximating Predictive Variances.

Obtaining the predictive variance based on the posterior of the weights associated to the support set is a bad idea, since those will be smaller the further away the test input is from the inputs in the support set. An explicit approximation to the predictive variance of a full GP, given in (3.6) is proposed instead. For a given test input $\mathbf{x}_*$, Smola and Bartlett (2001); Schölkopf and Smola (2002) propose to approximate the term:

$$-\mathbf{k}_*^\top \left[\mathbf{K} + \sigma^2 \mathbf{I}\right]^{-1} \mathbf{k}_* \ , \tag{3.36}$$

using the fact that it is the minimum (with respect to the $n \times 1$ weights vector $\boldsymbol{\beta}$, one weight associated to each training input) of the quadratic form:

$$-2\,\mathbf{k}_*^\top \boldsymbol{\beta} + \boldsymbol{\beta}^\top \left[\mathbf{K} + \sigma^2 \mathbf{I}\right] \boldsymbol{\beta} \ . \tag{3.37}$$

They then go on to propose finding a sparse version $\boldsymbol{\beta}_M$ of $\boldsymbol{\beta}$ with only M non-zero elements.[8] The method is again a greedy incremental minimisation of

[8]This M does not have anything to do with the number of inputs in the support set of our previous discussion. It corresponds to a *new* support set, this time for approximating

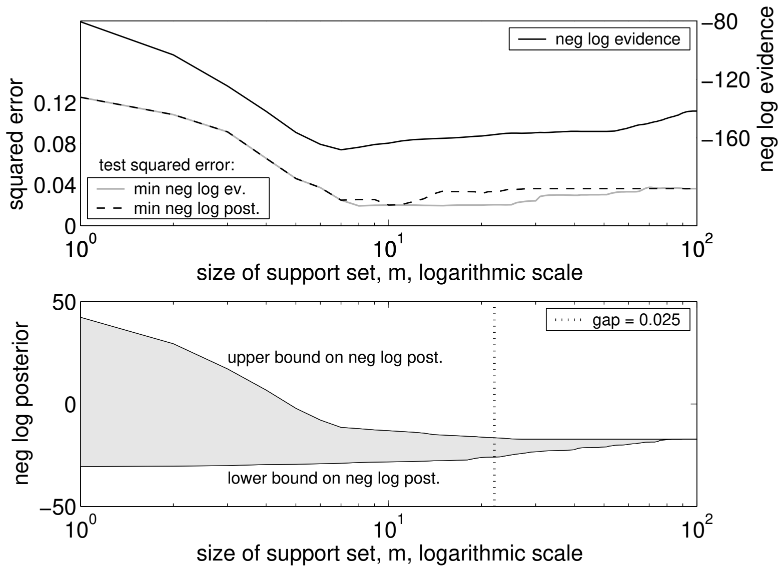

Figure 3.2: Comparison between a sparse greedy approximation based on minimising the negative log evidence, and one based on minimising the negative log posterior. In both figures the horizontal axis indicates the size of the support set. *Top:* the solid black curve is the negative log evidence, with values given by the right vertical axis, the other two curves are the test squared error of the greedy methods based on minimising the negative log evidence (solid gray) and the negative log posterior (dashed black), with values given on the left vertical axis. *Bottom:* for the SGGP approach the upper and lower bounds on the negative lower posterior are given, and the vertical dotted line shows the minimum size of the support set for which the "gap" is smaller that 0.025.

the expression in (3.37). For a given choice of active elements (non-zero) in $\boldsymbol{\beta}$, the minimum of the objective function is given by:

$$-\mathbf{k}(\mathbf{x}_*)^\top \left[\mathbf{K}_{MM} + \sigma^2\,\mathbf{I}\right]^{-1} \mathbf{k}(\mathbf{x}_*) \ , \tag{3.38}$$

where here again $\mathbf{k}(\mathbf{x}_*)$ represents an $M \times 1$ vector containing the covariance function evaluated at $\mathbf{x}_*$ and at the M inputs in the support set. Again, the support set yielding a minimal value of the expression in (3.38) will be chosen. The expression in (3.38) is also an upper bound on the (3.36), which means that bad approximations only mean an overestimate of the predictive variance, which is less bad than an underestimate. For each candidate to the

the predictive variance at $\mathbf{x}_*$. We insist on using the same symbol though because it still corresponds to a support set with $M < N$.

support set, (3.38) can be evaluated in $\mathcal{O}(M^2)$ (this cost includes updating $\left[\mathbf{K}_{MM} + \sigma^2\,\mathbf{I}\right]^{-1}$). Luckily, in practice the typical size of the support sets for approximating predictive variances is around one order of magnitude smaller than the size of the support set for approximating predictive means. Smola and Bartlett (2001); Schölkopf and Smola (2002) also provide a lower bound to (3.36), which allows to use a similar stop criterion as in the approximation of the predictive means.

3.3.5.4 Limitations

Though it does work in practice and for the datasets on which we have tried it, there is no fundamental guarantee that SGGP will always work, since it does not maximise the whole of the evidence: it ignores the term in $\log|\tilde{Q}|$.

The hyperparameters of the covariance function need to be known: they cannot be learned by maximising the posterior, since this would lead to over-fitting. For example, one would obtain zero lengthscales with the squared exponential covariance function.

While for approximating the predictive means one needs to find a unique support set, a specific support set needs to be estimated for *each* different test input if one wants to obtain good approximations to the predictive variance. The computational cost becomes then $\mathcal{O}(kNM^2)$ per training case, where k is the size of a reduced random search set (Smola and Bartlett (2001) suggest using $k = 59$).

3.4 Experiments

We use the KIN40K dataset (for more details see Rasmussen, 1996, Chap. 5). This dataset represents the forward dynamics of an 8 link all-revolute robot arm. The dataset contains 40000 examples, the input space is 8-dimensional, and the 1-dimensional output represents the distance of an end-point of the robot arm from a fixed point. The mapping to be learned is low noise and highly nonlinear. This is of importance, since it means that the predictions can be improved by training on more data, and sparse solutions do not arise trivially.

We divide the dataset into 10 disjoint subsets of 4000 elements, that we then further split into training and test sets of 2000 elements each. The size of the support set is set to 512 elements in all cases. For each method we perform then 10 experiments, and compute the following losses: the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Negative Test Log-density (NTL). We also compute the training negative log likelihood per training case. Averaged results over the 10 disjoint sub-datasets are shown in the upper part of Table 3.1. SGGP is the sparse support set selection method proposed by Smola and Bartlett (2001); to compute predictive uncertainties, we do not use the sparse greedy approximation they suggest, since it has a too high computational cost of $\mathcal{O}(kNM^2)$ per test case, with $k = 59$ and $M \approx 250$ in our case to reach $gap < 0.025$. As an alternative, they suggest to use the predictive uncertainties given by a reduced GP trained only on the support set obtained for approximating the predictive mean; the computational cost is low, $\mathcal{O}(M^2)$ per test case, but the performance is too poor to be worth reporting (NTL of the order of 0.3). To compute predictive uncertainties with the SGGP method we use the expressions given by (3.27) and (3.29). SGEV is our alternative greedy support set selection method based on maximising the evidence. The HPEV-rand method selects a support set at random and learns the covariance hyperparameters by maximising the evidence of the approximate model, as described in Sect. 3.3.1. The HPEV-SGEV and HPEV-SGGP methods select the support set for fixed hyperparameters according to the SGEV and and SGGP methods respectively, and then for that selected support set learn the hyperparameters by using HPEV. This procedure is iterated 10 times for both algorithms, which is enough for the likelihood to apparently converge. For all algorithms we present the results for the naïve non-augmented degenerate prediction model, and for the augmented non-degenerate one.

The experimental results show that the performance is systematically superior when using the augmented non-degenerate RRGP with an additional weight w_*. This superiority is expressed in all three losses, mean absolute, mean squared and negative test predictive density (which takes into account the predictive uncertainties). We believe that the relevant loss is the last one, since it reflects the fundamental theoretical improvement of the non-degenerate RRGP. The fact that the losses related to the predictive mean are also better can be explained by the model being slightly more flexible. We performed paired t-tests that confirmed that under all losses and algorithms considered, the augmented RRGP is significantly superior than the non-augmented one, with p-values always smaller than 1%. We found that for the dataset considered SGGP, SGEV and HPEV-rand are not significantly different. It would then seem that learning the hyperparameters for a random support set, or learning the support set for (carefully selected) hyperparameters by maximising the posterior or the evidence are methods with equivalent performance. We found that both for the augmented and the non-augmented case, HPEV-SGEV and HPEV-SGGP are significantly superior to the other three methods, under all losses, again with p-values below 1%. On the other hand, HPEV-SGEV and HPEV-SGGP are not significantly different from each other under any of the losses.

method	tr. neg ev.	*non-augmented* MAE	MSE	NTL	*augmented* MAE	MSE	NTL
SGGP	–	0.0481	0.0048	−0.3525	0.0460	0.0045	−0.4613
SGEV	−1.1555	0.0484	0.0049	−0.3446	0.0463	0.0045	−0.4562
HPEV-rand	−1.0978	0.0503	0.0047	−0.3694	0.0486	0.0045	−0.4269
HPEV-SGEV	−1.3234	0.0425	0.0036	−0.4218	0.0404	0.0033	−0.5918
HPEV-SGGP	−1.3274	0.0425	0.0036	−0.4217	0.0405	0.0033	−0.5920
		2000 *training* - 2000 *test*					
SGEV	−1.4932	0.0371	0.0028	−0.6223	0.0346	0.0024	−0.6672
HPEV-rand	−1.5378	0.0363	0.0026	−0.6417	0.0340	0.0023	−0.7004
		36000 *training* - 4000 *test*					

Table 3.1: Comparison of different learning methods for RRGPs on the KIN40K dataset, for 2000 training and test cases (*upper subtable*) and for 36000 training and 4000 test cases (*lower subtable*). The support set size is set to 512 for all methods. For each method the training negative log marginal likelihood per case is given, together with the Mean Absolute Error (MAE), Mean Squared Error (MSE) and Negative Test Log-likelihood (NTL) losses. SGGP (Smola and Bartlett, 2001) and SGEV (our alternative to SGGP based on maximising the evidence) are based on learning the support set for fixed hyperparameters. HPEV-random learns the hyperparameters for a random subset, and HPEV-SGEV and HPEV-SGGP are methods where SGEV and SGGP are respectively interleaved with HPEV, for 10 repetitions.

The lower part of Table 3.1 shows the results of an additional experiment we made, where we compare SGEV to HPEV-rand on a larger training set. We generate this time 10 disjoint test sets of 4000 cases, and 10 corresponding training sets of 36000 elements. The size of the support sets remains 512. We compute the same losses as earlier, and consider also the augmented and the non-augmented RRGPs for making predictions. Paired t-tests[9] confirm once again the superiority of the augmented model to the non-augmented one for both models and all losses, with p-values below 1%.

3.5 Discussion

We have proposed to augment RRGPs at test time, by adding an additional weight w_* associated to the new test input $\mathbf{x}_*$. The computational cost for the predictive mean increases to $\mathcal{O}(NM)$ per case, i.e. $\mathcal{O}(N)$ more expensive

[9]Due to dependencies between the training sets, assumptions of independence needed for the t-test could be compromised, but this is probably not a major effect.

than the non-augmented case. It might seem surprising that this is more expensive than the $\mathcal{O}(N)$ cost per case of the full GP! Of course, the full GP has has an initial cost of $\mathcal{O}(N^2)$ provided that the covariance matrix has been inverted, which costs $\mathcal{O}(N^3)$. Computing predictive variances has an initial cost of $\mathcal{O}(NM^2)$ like for the non-augmented case, and then a cost per case of $\mathcal{O}(NM)$ which is more expensive than the $\mathcal{O}(M^2)$ for the non-augmented case, and below the $\mathcal{O}(N^2)$ of the full GP. It may be argued that the major improvement brought by augmenting the RRGP is in terms of the predictive variance, and that one might therefore consider computing the predictive mean from the non-augmented model, and the predictive variance from the augmented. However, the experiments we have conducted show that the augmented RRGP is systematically superior to the non-augmented, for all losses and learning schemes considered. The mean predictions are also better, probably due to the gain in flexibility by having an additional basis function.

Which method should be used for computing predictive variances? We have shown that using the degenerate RRGP, (3.27), has a computational cost of $\mathcal{O}(M^2)$ per test case. Using the augmented non-degenerate RRGP is preferable though because it gives higher quality predictive uncertainties, but the cost augments to $\mathcal{O}(NM)$ per test case. Smola and Bartlett (2001) propose two possibilities. A cost efficient option, $\mathcal{O}(M^2)$ per test case, is to base the calculation of all test predictive variances on the support set selected by approximating the posterior, which is in fact equivalent to computing predictive variances from a small full GP trained only on the support set. They show that the predictive variances obtained will always be an upper bound on the ones given by the full GP, and argue that inaccuracy (over estimation) is for that reason benign. We found experimentally that the errorbars from a small full GP trained only on the support set are very poor. The more accurate, yet more costly option consists is selecting a new support set for each test point. While they argue that the typical size of such test sets is very small (of the order of 25 for reasonable hyperparameters for the abalone dataset, but of the order of 250 for the KIN40K dataset), the computational cost per test case rises to $\mathcal{O}(kNM^2)$. As we have explained, k is the size of a reduced random search set that can be fixed to 59 (see Smola and Bartlett, 2001). For their method to be computationally cheaper than our augmented RRGP, the support set that our method selects should contain more than $59 \times 25^2 = 36875$ elements. This is two orders of magnitude above the reasonable size of support sets that we would choose. In the experiments, we ended up computing the predictive variances for the SGGP from our expressions (3.27) and (3.29).

We found that none of the two possible "one-shot" approaches to training a RRGP is significantly superior to the other. In other words, selecting support sets at random and optimising the hyperparameters does not provide significantly different performance than fixing the hyperparameters and selecting the support set in a supervised manner. Furthermore, on the dataset we did our experiments SGGP and SGEV did not prove to be significantly different either. We expect SGEV to perform better than SGGP on datasets where for the given hyperparameters the learning curve saturates, or even deteriorates as the support set is increased, as is the case in the example we give in Fig. 3.2. Interleaving support set selection and hyperparameter learning schemes proves on the other hand to be promising. The experiments on KIN40K show that this scheme gives much superior performance to the two isolated learning schemes.

It is interesting to note the relation between the RRGP and the Nyström approximation proposed by Williams and Seeger (2001). In that approach the predictive mean and variance are respectively given by:

$$\begin{aligned} m(\mathbf{x}_*) &= \mathbf{k}_*^\top \left[\mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\,\mathbf{K}_{NM}^\top + \sigma^2\,\mathbf{I}\right]^{-1} \mathbf{y} \ , \\ v(\mathbf{x}_*) &= \sigma^2 + k_{**} + \mathbf{k}_*^\top \left[\mathbf{K}_{NM}\,\mathbf{K}_{MM}^{-1}\,\mathbf{K}_{NM}^\top + \sigma^2\,\mathbf{I}\right]^{-1} \mathbf{k}_* \ . \end{aligned} \tag{3.39}$$

These expressions are very similar to those obtained for the augmented RRGP, given by (3.29). However, the additional term in the approximate covariance for the augmented RRGP ensures that it is positive definite, see (Williams et al., 2002), and that therefore our approach does not suffer from negative predictive variances as is the case for the Nyström approximation for GPs.

We are currently working on a more exhaustive literature review of sparse GPs, that will include experimental comparisons between the different approaches that have recently been proposed. It will be interesting to see whether the experiments suggest the existence of any clearly superior paradigm for sparse GPs.

CHAPTER 4

Uncertainty in the Inputs

When presented with pairs of inputs and outputs at training time, or with inputs only a test time, we have until now made the extremely common assumption that only the outputs are noisy, and we have explicitly modelled the output noise. Consider first the situation where it is acceptable to think of the training inputs as deterministic, but where the same cannot be done with the test inputs. An example of such situation is the case of iterative time-series predictions with GPs, discussed in Sect. 4.2, where the inputs are composed of previous predictions. Since GPs produce predictive distributions, we know that the inputs to our model are random, and we know their distribution. When predicting k-steps ahead, we rely on $k-1$ intermediate predictions, all of which uncertain. Failing to take into account this accumulated uncertainty implies that the predictive distribution of the k-th prediction is very overconfident. Addressing this problem raises the issue of how to predict with a GP at an uncertain input with known distribution. In Sect. 4.1 we derive the equations for computing a Gaussian approximation to the predictive distribution at an uncertain input, with known Gaussian distribution, for GPs and for RVMs. Indeed, unless the model is linear, Gaussian input distributions are mapped by the non-linearities into arbitrary predictive distributions, hence the need to approximate these by a Gaussian. This need is also motivated by the fact that in propagating the uncertainty in iterative time-series predictions, we will want to always have inputs to the GP with normal distributions.

The issue of how to train a GP when the training inputs are noisy has proven to be more challenging. We present our attempts to addressing it in Sect. 4.3, where we first provide a brief enumeration of different ways of dealing with learning with uncertain inputs proposed in the literature. Under a Bayesian perspective, one would want to integrate over the uncertain inputs. As is often the case, this integration is analytically intractable, and we propose two approximations. The first consists in getting rid of the integral altogether by finding the maximum of the joint posterior over uncertain inputs and GP hyperparameters. This proves to be a challenging optimisation problem, with very many undesirable spurious optima that lead to over-fitting. The optimisation can be made practical by using an annealing approach, where instead of estimating the output noise we gradually reduce it while learning the remaining parameters. A feature of this method is that it effectively performs an *imputation* of the "true" inputs, which may be of interest in its own right. Unfortunately no obvious stopping criterion has been found for the annealing procedure. Knowledge of the actual output noise level is required, which is not satisfactory. In our second approach to learning GPs with input noise, we propose to iteratively sample from the posterior on uncertain inputs, and learn the hyperparameters by maximising their posterior. This can naturally be cast as a "stochastic" Expectation-Maximisation (EM) algorithm. While this approach allows us to learn the output noise, it increases the already high computational cost of training a GP model, and is therefore impractical for datasets larger than a few hundred samples.

4.1 Predicting at an Uncertain Input

Suppose we have trained our model, GP or RVM, on a training set with inputs $\{\mathbf{x}_i | i = 1, \dots, N\} \subset \mathbb{R}^D$ organised as rows in matrix X, and corresponding targets $\mathbf{y} = [y_1, \dots, y_N]^\top$. Let θ represent here the set of learned parameters (e.g. lengthscales and output noise variance). For a deterministic test input $\mathbf{x}_*$, the predictive distribution of the function value $p(f_*|\mathbf{y}, \mathbf{x}_*, X, \theta)$ ($f_* = f(\mathbf{x}_*)$ for simplicity) is Gaussian with mean and variance given for GPs by (3.6), and for RVMs by (2.10). For convenience we will reproduce these expressions here, in a slightly different form. For a GP with covariance function $K(\cdot,\cdot)$, the predictive mean and variance are given by:

$$m(\mathbf{x}_*) = \sum_{i=1}^{N} \beta_i \, K(\mathbf{x}_*, \mathbf{x}_i) \;\; , \qquad v(\mathbf{x}_*) = k_{**} - \sum_{i,j=1}^{M} \mathbf{\Omega}_{ij} \, K(\mathbf{x}_*, \mathbf{x}_i) \, K(\mathbf{x}_*, \mathbf{x}_j) \;\; , \tag{4.1}$$

where we have defined $\mathbf{\Omega} \equiv (\mathbf{K} + \sigma^2 \, \mathbf{I})^{-1}$, where $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ is the covariance matrix of the GP, σ^2 is the estimated output noise variance, and $\mathbf{I}$ is the identity matrix. We also have defined β_i as the i-the element of column vector $\boldsymbol{\beta} = \mathbf{\Omega}\, \mathbf{y}$. For RVMs, the predictive mean and variance are given by:

$$m(\mathbf{x}_*) = \sum_{i=1}^{M} \mu_i\, \phi_i(\mathbf{x}_*) \ , \qquad v(\mathbf{x}_*) = \sum_{i,j=1}^{M} \mathbf{\Sigma}_{ij}\, \phi_i(\mathbf{x}_*)\, \phi_j(\mathbf{x}_*) \ , \tag{4.2}$$

where M is the number of Relevance Vectors (RVs), and $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_N]^\top$ and $\mathbf{\Sigma}$ are the mean and the covariance of the weights posterior, (2.19) and (2.18). $\phi_i(\mathbf{x}_*) \subset [\mathbb{R}^D \to \mathbb{R}]$ is the $i - th$ basis function.

Consider now the situation where the test input is random, with Gaussian input distribution, $\mathbf{x}_* \sim \mathcal{N}(\mathbf{u}, \mathbf{S})$, with known mean and variance. The new, marginal predictive distribution of f_* is obtained by integrating over the input distribution:

$$p(f_*|\mathbf{u}, \mathbf{S}) = \int p(f_*|\mathbf{x}_*)\, p(\mathbf{x}_*|\mathbf{u}, \mathbf{S})\, \mathrm{d}\mathbf{x}_* \ , \tag{4.3}$$

where for simplicity we will from now on write the predictive distribution as $p(f_*|\mathbf{x}_*)$ and the marginal predictive distribution as $p(f_*|\mathbf{u}, \mathbf{S})$, omitting to explicitly condition on $\mathbf{y}$, X and θ. We call this new predictive distribution "marginal" because it is obtained by marginalising the predictive distribution with respect to the input $\mathbf{x}_*$ and the conditioning is now on the parameters of the input distribution $\mathbf{u}$ and $\mathbf{S}$. For most covariance functions and basis functions, the predictive distribution depends highly non-linearly on $\mathbf{x}_*$, which on the one hand makes the analytic integration impossible, and on the other hand implies that the marginal predictive distribution is not Gaussian anymore, and can probably not easily be parametrised in terms of standard distributions. This effect is illustrated in Fig. 4.1, where we have used sampling and then smoothened with a Parzen estimator to obtain an estimate of the marginal predictive distribution, which is in this case multi-modal. In the same figure we show the mean and variance of two different Gaussian approximations to the marginal predictive distribution, which we describe in what follows.

One obvious possibility for approximating the integral in (4.3) is to use a simple Monte-Carlo approach:[1]

$$p(f_*|\mathbf{u}, \mathbf{S}) \simeq \frac{1}{\mathrm{T}} \sum_{t=1}^{\mathrm{T}} p(f_*|\mathbf{x}_*^t) \ , \tag{4.4}$$

where $\{\mathbf{x}_*^t | t = 1 \ldots, \mathrm{T}\}$ are independent samples from $p(\mathbf{x}_*|\mathbf{u}, \mathbf{S})$, which is very easy to sample from since it is Gaussian. It might be advantageous however to have a parametric approximation to the marginal predictive distribution, which

[1] This is how we obtained the "sampling" approximation in Fig. 4.1

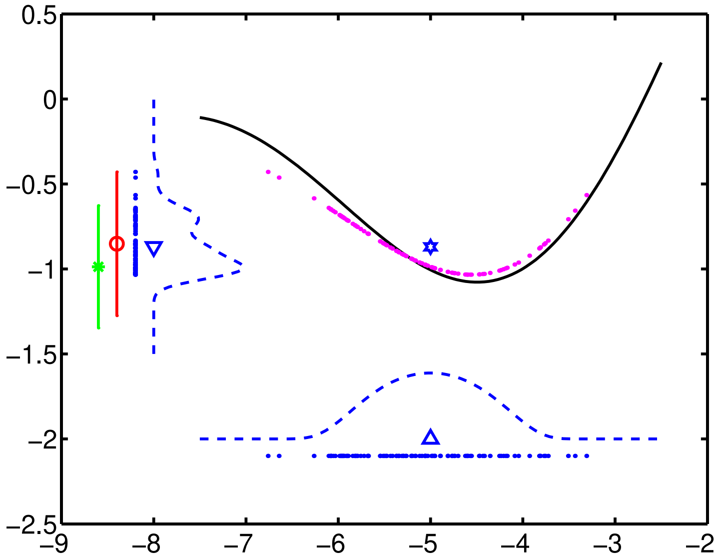

Figure 4.1: Prediction with uncertain input. *On the x-axis*, the dashed line represents the Gaussian input distribution, with mean located by the triangle, from which we draw 100 samples (dots under it). *In the middle of the figure*, the solid line represents the true underlying function. We fit a model to it, and propagate the 100 input samples through the model (dots close to the true function). *On the y-axis* we project the 100 predicted values (dots) and use them to estimate the predictive density (dashed line), with mean located by the triangle. The error bar with a circle and the error bar with a star show the mean and 95% confidence interval of the Gaussian approximation with exact computation of mean and variance and of the method with Taylor expansion respectively.

is why we settle for projecting it onto a Gaussian distribution. We will therefore now be concerned with computing its mean and variance. These can be obtained using respectively the law of iterated expectations and law of conditional variances:

$$\begin{aligned} m(\mathbf{u},\mathbf{S}) &= \mathrm{E}_{f_*}\left[f_*\,\mathrm{E}_{\mathbf{x}_*}[p(f_*|\mathbf{x}_*)]\right] = \mathrm{E}_{\mathbf{x}_*}\left[\mathrm{E}_{f_*}[f_*\,p(f_*|\mathbf{x}_*)]\right] \\ &= \mathrm{E}_{\mathbf{x}_*}[m(\mathbf{x}_*)]\ , \end{aligned} \tag{4.5}$$

$$\begin{aligned} v(\mathbf{u},\mathbf{S}) &= \mathrm{E}_{\mathbf{f}_*}\left[f_*^2\,\mathrm{E}_{\mathbf{x}_*}[p(f_*|\mathbf{x}_*)]\right] - \left(\mathrm{E}_{\mathbf{f}_*}\left[f_*\,\mathrm{E}_{\mathbf{x}_*}[p(f_*|\mathbf{x}_*)]\right]\right)^2 \\ &= \mathrm{E}_{\mathbf{x}_*}\left[\mathrm{E}_{\mathbf{x}_*}[f_*^2 p(f_*|\mathbf{x}_*)]\right] - \left(\mathrm{E}_{\mathbf{x}_*}\left[\mathrm{E}_{\mathbf{x}_*}[f_*\,p(f_*|\mathbf{x}_*)]\right]\right)^2 \\ &= \mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*) + m(\mathbf{x}_*)^2] - \mathrm{E}_{\mathbf{x}_*}[m(\mathbf{x}_*)]^2 \\ &= \mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*)] + \mathrm{Var}_{\mathbf{x}_*}[m(\mathbf{x}_*)]\ , \end{aligned} \tag{4.6}$$

where $\mathrm{E}_{\mathbf{x}_*}$ and $\mathrm{Var}_{\mathbf{x}_*}$ indicate respectively the expectation and the variance under $p(\mathbf{x}_*|\mathbf{u},\mathbf{S})$, and $\mathrm{E}_{\mathbf{f}_*}$ and Var_{f_*} the expectation and variance under $p(f_*|\mathbf{x}_*)$.

The analytical difficulties are however still present, since we now need to compute the terms $E_{\mathbf{x}_*}[m(\mathbf{x}_*)]$, $E_{\mathbf{x}_*}[m(\mathbf{x}_*)^2]$ and $\mathrm{E}_{x_*}[v(\mathbf{x}_*)]$. For the general case, one possibility is to perform a Taylor expansion of $m(\mathbf{x}*)$ and $v(\mathbf{x}_*)$ about the mean of the input distribution, $\mathbf{u}$. No matter what the order of the approximation is, things become analytically tractable since the integrals are over polynomials times Gaussians. In (Girard et al., 2003) this approximation is described, and in a more extended manner in (Girard et al., 2002). We will refer to the computation of the mean and variance of the predictive distribution as the "approximate" moment matching method, as opposed to the "exact" moment matching method that we describe immediately.

4.1.1 Exact Moment Matching

In (Quiñonero-Candela et al., 2003a) we made the observation that for certain types of covariance functions and basis functions exact computation of the mean and variance of the predictive distribution are possible. In particular, we derived the expressions for the particular yet fairly common case of squared exponential covariance and basis functions, (3.1) and (2.6) and reproduced below in (4.7), and we will in this chapter restrict ourselves to this case only. The details of this derivation were relegated to (Quiñonero-Candela et al., 2003b). Girard (2004) has derived the expressions for exact moment matching for other forms of covariance functions, like the linear case.

For convenience, let us reproduce here the expression of a squared exponential covariance function:

$$K(\mathbf{x}_i,\mathbf{x}_j) = \theta_{D+1}^2\,\exp\left(-\frac{1}{2}(\mathbf{x}_i-\mathbf{x}_j)^\top\,\mathbf{\Theta}^{-1}(\mathbf{x}_i-\mathbf{x}_j)\right)\ , \tag{4.7}$$

where θ_{D+1} relates to the amplitude of the functions generated by the GP, and $\mathbf{\Theta}$ is a diagonal matrix whose elements $\{\theta_d | d = 1, \ldots, D\}$ are the lengthscales in the d-th dimensions. For the RVM, the basis functions are given by $\phi_i(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$ with $\theta_{D+1} = 1$. We will perform the derivations explicitly (even verbosely) only for the GP case. Since those for RVMs are very similar, we will only give the final expressions.

Computing the mean. From (4.5) and (4.1) have:

$$m(\mathbf{u}, \mathbf{S}) = \int m(\mathbf{x}_*)\, p(\mathbf{x}_* | \mathbf{u}, \mathbf{S})\, \mathrm{d}\mathbf{x}_* = \sum_{i=1}^{N} \beta_i\, l_i = \boldsymbol{\beta}^\top \boldsymbol{l} \ , \tag{4.8}$$

where $\boldsymbol{l}$ is a column vector with elements l_i, that are given by:

$$l_i = \int K(\mathbf{x}_*, \mathbf{x}_i)\, p(\mathbf{x}_* | \mathbf{u}, \mathbf{S})\, \mathrm{d}\mathbf{x}_* \ . \tag{4.9}$$

Computing l_i is a simple task, since it is the integral in $\mathbf{x}_*$ of the product of two Gaussians in $\mathbf{x}_*$ except for a normalisation constant. Indeed $K(\mathbf{x}_*, \mathbf{x}_i) = \mathrm{Z}_K\, \mathcal{N}(\mathbf{x}_i, \mathbf{\Theta})$ where the normalisation constant is $\mathrm{Z}_K = \theta_{D+1}^2\, (2\pi)^{D/2} |\mathbf{\Theta}|^{1/2}$. Using the algebra provided in App. A.2 (with $P = \mathbf{I}$), we obtain:

$$\begin{aligned} l_i =& \theta_{D+1}^2 |\mathbf{I} + \mathbf{S}\, \mathbf{\Theta}^{-1}|^{-1/2} \exp\left(-\frac{1}{2} (\mathbf{u} - \mathbf{x}_i)^\top (\mathbf{\Theta} + \mathbf{S})^{-1} (\mathbf{u} - \mathbf{x}_i) \right) \\ =& K(\mathbf{u}, \mathbf{x}_i)\, |\mathbf{I} + \mathbf{S}\, \mathbf{\Theta}^{-1}|^{-1/2} \\ & \cdot \exp\left(\frac{1}{2} (\mathbf{u} - \mathbf{x}_i)^\top (\mathbf{\Theta} + \mathbf{S})^{-1} \mathbf{S} \mathbf{\Theta}^{-1} (\mathbf{u} - \mathbf{x}_i) \right) \ . \end{aligned} \tag{4.10}$$

Interestingly, the new mean of the marginal predictive distribution is very similar to the mean of the predictive distribution when predicting at $\mathbf{u}$. Indeed, it is a weighted sum, with the same weight vector $\boldsymbol{\beta}$ as for the deterministic case, of evaluations of a squared exponential covariance function. This new covariance function is given by $\tilde{K}(\mathbf{u}, \mathbf{x}_i) = l_i$. It is easy to see that if $\mathbf{x}_* = \mathbf{u}$ is certain, and therefore $\mathbf{S}$ is the zero matrix, then $\tilde{K}(\mathbf{u}, \mathbf{x}_i) = K(\mathbf{u}, \mathbf{x}_i)$ and the marginal predictive mean is equivalent to the predictive mean at $\mathbf{u}$. In the case where $\mathbf{x}_*$ is uncertain, to compare both covariance functions let us write the following function of $\mathbf{v} = \mathbf{u} - \mathbf{x}_i$:

$$g(\mathbf{v}) = \log \frac{\tilde{K}(\mathbf{u}, \mathbf{x}_i)}{K(\mathbf{u}, \mathbf{x}_i)} = \mathbf{v}^\top (\mathbf{\Theta} + \mathbf{S})^{-1} \mathbf{S} \mathbf{\Theta}^{-1}\, \mathbf{v} - \log |\mathbf{S}\, \mathbf{\Theta}^{-1} + \mathbf{I}| \ . \tag{4.11}$$

We see that $g(\mathbf{v})$ is convex. When $\mathbf{v} = \mathbf{0}$, we have $g(\mathbf{v}) < 0$.[2] From convexity arguments, for any $\mathbf{v}$ in the region inside the ellipse centred at $\mathbf{0}$ defined by

[2] This is because $|\mathbf{S}\, \mathbf{\Theta}^{-1} + \mathbf{I}| > 1$, which we get from the fact that for $\mathbf{A}$ and $\mathbf{B}$ positive definite $|\mathbf{A} + \mathbf{B}| \geq |\mathbf{A}| + |\mathbf{B}|$.

$g(\mathbf{v}) = 0$, we have $g(\mathbf{v}) < 0$ which implies $\tilde{K}(\mathbf{u}, \mathbf{x}_i) < K(\mathbf{u}, \mathbf{x}_i)$. Furthermore, we roughly see that the radius of the ellipse increases with the ratio of the determinant of $\mathbf{S}$ to that of $\boldsymbol{\Theta}$. We deduce that when computing the marginal predictive mean, the more uncertain $\mathbf{x}_*$ is with respect to the covariance lengthscales, the less weight the targets of the inputs close to $\mathbf{u}$ get and the more those that are far. This has a smoothing effect, analogous to that of increasing the lengthscales in a standard GP. It is not surprising to get smoother functions if the inputs are uncertain. Compared to the first order Taylor approximation used in (Girard et al., 2003), we do not have anymore that $m(\mathbf{u}, S) = m(\mathbf{u})$, i.e. the marginal predictive mean is not anymore the predictive mean at $\mathbf{u}$; under that approach there was no smoothing of the predictive mean.

For the RVM, the marginal predictive mean is given by:

$$m(\mathbf{u}, \mathbf{S}) = \boldsymbol{\mu}^\top \boldsymbol{l} \ , \tag{4.12}$$

where for computing $\boldsymbol{l}$ we have set $K(\mathbf{u}, \mathbf{x}_i) = \phi_i(\mathbf{u})$.

Computing the variance. From (4.6) we see that the marginal predictive variance is the sum of the expected marginal variance and the variance of the predictive mean, $v(\mathbf{u}, \mathbf{S}) = \mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*)] + \mathrm{Var}_{\mathbf{x}_*}[m(\mathbf{x}_*)]$. Using (4.1) we see that these two terms are respectively given by:

$$\begin{aligned} \mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*)] &= \theta_{D+1}^2 - \sum_{i,j=1}^{N} \boldsymbol{\Omega}_{ij}\, L_{ij} \ = \ \theta_{D+1}^2 - \mathrm{Tr}[\boldsymbol{\Omega}\, \mathbf{L}] \ , \\ \mathrm{Var}_{\mathbf{x}_*}[m(\mathbf{x}_*)] &= \mathrm{E}_{\mathbf{x}_*}[m(\mathbf{x}_*)^2] - \mathrm{E}_{\mathbf{x}_*}[m(\mathbf{x}_*)]^2 \\ &= \sum_{i,j=1}^{N} \beta_i \beta_j\, L_{ij} - [\boldsymbol{\beta}^\top \boldsymbol{l}]^2 \ = \ \mathrm{Tr}[\boldsymbol{\beta}\boldsymbol{\beta}^\top (\mathbf{L} - \boldsymbol{l}\,\boldsymbol{l}^\top)] \ , \end{aligned} \tag{4.13}$$

where $\mathrm{E}_{\mathbf{x}_*}[m(\mathbf{x}_*)]^2$ is readily obtained from (4.8), and matrix $\mathbf{L}$ has elements L_{ij} given by:

$$L_{ij} = \int K(\mathbf{x}_*, \mathbf{x}_i)\, K(\mathbf{x}_*, \mathbf{x}_j)\, p(\mathbf{x}_* | \mathbf{u}, \mathbf{S})\, \mathrm{d}\mathbf{x}_* \ . \tag{4.14}$$

This integral is the product of three Gaussians in $\mathbf{x}_*$. To see how we can compute it, let us start by writing the product of the two covariance evaluations, which is again proportional to Gaussian in $\mathbf{x}_*$:

$$K(\mathbf{x}_*, \mathbf{x}_i)\, K(\mathbf{x}_*, \mathbf{x}_j) = \mathrm{Z}_{KK} \mathcal{N}(\mathbf{x}_d, \boldsymbol{\Theta}/2) \ , \tag{4.15}$$

where we have defined $\mathbf{x}_d = (\mathbf{x}_i + \mathbf{x}_j)/2$, and where the normalisation constant is given by:

$$\mathrm{Z}_{KK} = \mathrm{Z}_K^2 (2\pi)^{-D/2} |2\boldsymbol{\Theta}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^\top (2\boldsymbol{\Theta})^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right) \ . \quad (4.16)$$

We have thus reduced the integrand in (4.14) to the product of two Gaussians in $\mathbf{x}_*$. The integral is thus the corresponding normalising constant, that can be computed using the algebra in App.A.2, and allows us to obtain:

$$\begin{aligned} L_{ij} =& \mathrm{Z}_{KK}\, (2\pi)^{-D/2}\, |\boldsymbol{\Theta}/2 + \mathbf{S}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{u} - \mathbf{x}_d)^\top \left(\boldsymbol{\Theta}/2 + \mathbf{S}\right)^{-1} (\mathbf{u} - \mathbf{x}_d)\right) \\ =& K(\mathbf{u}, \mathbf{x}_i)\, K(\mathbf{u}, \mathbf{x}_j)\, |\mathbf{I} + 2\mathbf{S}\boldsymbol{\Theta}^{-1}|^{-1/2} \\ & \cdot \exp\left(\frac{1}{2}(\mathbf{u} - \mathbf{x}_d)^\top (\boldsymbol{\Theta}/2 + \mathbf{S})^{-1} \mathbf{S}\, (\boldsymbol{\Theta}/2)^{-1} (\mathbf{u} - \mathbf{x}_d)\right) \ . \end{aligned} \quad (4.17)$$

We could now perform an analysis similar to previously, where we compare L_{ij} to $K(\mathbf{u}, \mathbf{x}_i)\, K(\mathbf{u}, \mathbf{x}_j)$. We see that if $\mathbf{x}_* = \mathbf{u}$ is certain, $\mathbf{S}$ is the zero matrix and the two quantities are equal. Let us consider the case where $\mathbf{S}$ is not zero. If we define this time $\mathbf{v} = \mathbf{u} - \mathbf{x}_d$, and take the logarithm of the ratio between L_{ij} to $K(\mathbf{u}, \mathbf{x}_i)\, K(\mathbf{u}, \mathbf{x}_j)$ and proceed as in (4.11), we again obtain that there exists an ellipse in $\mathbf{v}$-space (with radius that grows with $\mathbf{S}$) such that if $\mathbf{v}$ is inside the ellipse we have $L_{ij} < K(\mathbf{u}, \mathbf{x}_i)\, K(\mathbf{u}, \mathbf{x}_j)$, and the opposite if $\mathbf{v}$ is outside the ellipse. This implies that $\mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*)]$ will be smaller than $\mathbf{v}(\mathbf{u})$ when $\mathbf{v}$ is large compared to $\mathbf{S}$.

One might have initially thought that $\mathbf{x}_*$ becoming random, the marginal predictive variance would become systematically larger than the predictive variance at its mean, $v(\mathbf{u}, \mathbf{S}) \geq v(\mathbf{u})$. We now can understand how this is not necessarily the case. Suppose we were in a region where the function was extremely flat, so that there would be no contribution from the variance of the mean $\mathrm{Var}_{\mathbf{x}_*}[m(\mathbf{x}_*)]$. Suppose in addition, that $\mathbf{u}$ was far away from the training points. We have seen that in such case we would actually have $\mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*)] < v(\mathbf{x}_*)$, which would result in a smaller marginal predictive variance: $v(\mathbf{u}, \mathbf{S}) < v(\mathbf{u})$. Outside from these particular situations, we have in practice experienced that the marginal predictive variance of $\mathbf{x}_*$ is generally greater than the predictive variance at $\mathbf{u}$.

It is also reassuring to notice that for the case where $\mathbf{S}$ tends to zero, L_{ij} tends to $K(\mathbf{u}, \mathbf{x}_i)\, K(\mathbf{u}, \mathbf{x}_j)$ and l_i tends to $K(\mathbf{u}, \mathbf{x}_i)$. This implies that $\mathrm{Var}_{\mathbf{x}_*}[m(\mathbf{x}_*)]$ tends to zero, and that $\mathrm{E}_{\mathbf{x}_*}[v(\mathbf{x}_*)]$ tends to $v(\mathbf{u})$. Therefore as $\mathbf{x}_*$ tends to be deterministic and equal to $\mathbf{u}$, $v(\mathbf{u}, \mathbf{S})$ tends to $v(\mathbf{u})$. For $\mathbf{S}$ equals zero, $v(\mathbf{u}, \mathbf{S}) = v(\mathbf{u})$, as one had hoped.

For the RVM, the marginal predictive variance is given by:

$$v(\mathbf{u}, \mathbf{S}) = \mathrm{Tr}[\boldsymbol{\Sigma}\mathbf{L}] + \mathrm{Tr}[\boldsymbol{\mu}\boldsymbol{\mu}^\top(\mathbf{L} - \boldsymbol{l}\boldsymbol{l}^\top)] \ , \tag{4.18}$$

where again, for computing $\boldsymbol{l}$ and $\mathbf{L}$ we have set $K(\mathbf{u}, \mathbf{x}_i) = \phi_i(\mathbf{u})$. Notice that $\mathbf{L}$ appears in the expected variance, $\mathrm{Tr}[\boldsymbol{\Sigma}\mathbf{L}]$, with opposite sign as for the GP case. This suggests that the inappropriate behaviour of the predictive variance that we discussed in Sect. 2.5 persists for the marginal predictive variance. It would probably be preferable to derive the equations for the marginal predictive mean and variance for the RVM* (Sect. 2.5).

4.2 Propagation of the Uncertainty

In Sect. 2.3 we described the use of RVMs for non-linear time-series predictions. In that setting we did limit ourselves to predicting at a fixed horizon, concretely six steps ahead, and we only interested ourselves for the predictive mean. There are many situations however, in which one may want to predict multiple steps ahead, like for example finance and control, and be able to give good estimates of the predictive uncertainty. Multiple step ahead time-series predictions can typically be performed under two approaches. The first approach consists in training the model to learn to predict on a fixed horizon of interest (direct method) and the second in training the model to learn to predict on a short horizon, and in reaching the horizon of interest by making repetitive one-step ahead predictions (iterative method). Farmer and Sidorowich (1988) conclude that iterative forecasting usually is superior to direct forecasting. The direct method has the disadvantages that as the forecast horizon increases, the complexity of the non-linear mapping increases as well, and the number of available input-output training pairs decreases. For the iterative method, the complexity of the non-linear mapping is much lower, and the model only needs to be trained once no matter what the forecast horizon of interest is. The disadvantage of the iterative method is that as the forecast horizon increases, the performance is diminished by the accumulated uncertainty of the intermediate predictions.

In this section we interest ourselves for the multiple step ahead case, and in particular for being able to take into account the fact that uncertain predictions are being fed back into the model as inputs. We will exploit our ability to predict at an uncertain input to propagate the uncertainty in multiple step ahead predictions, and obtain more appropriate predictive variances. Naïve iterative methods do not account for the accumulated uncertainty in the predictive distribution at a given horizon and soon become overconfident about the predictions.

4.2.1 Naïve Iterative k-step Ahead Prediction

Consider the discrete time series given by a set $\{y_t\}$ of samples ordered according to is an integer index t, and where the sampling period is constant. We model the time-series with a non-linear autoregressive model, as we did in Sec. 2.3:

$$\begin{cases} \mathbf{x}_t = [y_{t-1}, \ldots, y_{t-\tau}]^\top \\ y_t = f(\mathbf{x}_t) + \epsilon \end{cases} , \tag{4.19}$$

where the input $\mathbf{x}_t$ associated to time t is composed of previous outputs, up to a given lag[3] τ and we have an additive (white) noise with variance σ_ϵ^2. For notational convenience, we will add this noise variance to the expressions for the predictive variance and marginal predictive variance: $v(\mathbf{u}, \mathbf{S}) = v(\mathbf{u}, \mathbf{S}) + \sigma_\epsilon^2$. This expressions give us now the predictive variance of noisy targets y_t directly (instead of that of the associated noiseless function output $f(\mathbf{x}_t)$), and this is convenient for us, since we will always be considering the noisy targets in what follows.

Suppose the data is known up to time step T. The training set is composed by the input-target pairs $\{\mathbf{x}_t, y_t | t = 1, \ldots, T\}$. Suppose we want to predict the value of the time series at time $T + k$, i.e. k steps ahead. The first thing we do is to form the input vector $\mathbf{x}_{T+1} = [y_T, y_{T-1}, \ldots, y_{T+1-\tau}]^\top$, and since it is deterministic we can use the expressions for the mean and variance of the (standard) predictive distribution to obtain $y_{T+1} \sim \mathcal{N}(m(\mathbf{x}_T), v(\mathbf{x}_T))$. We now want to proceed to predict at y_{T+2}, for which we form the input vector $\mathbf{x}_{T+2} = [y_{T+1}, y_{T-1}, \ldots, y_{T+2-\tau}]^\top$. Now this vector contains the stochastic element y_{T+1} whose distribution we know, since we have just computed it. If one was not able to predict at uncertain inputs, one would resort to the naïve approach consisting in replacing the stochastic element by its mean $\hat{y}_{T+1} = m(\mathbf{x}_{T+1})$, and then using again the standard predictive mean and variance to obtain y_{T+2}. Let us give a schematic impression of how the process evolves, where we place hats on the variables that have been approximated by their

[3]We are not concerned with the identification of the lag and assume it has a known, fixed value.

mean:

$$
\begin{aligned}
\mathbf{x}_{T+1} = [y_T, y_{T-1}, \ldots, y_{T+1-\tau}]^\top \quad &\rightarrow \quad y_{T+1} \sim \mathcal{N}(m(\mathbf{x}_{T+1}), v(\mathbf{x}_{T+1})) \\
&\qquad \hat{y}_{T+1} = m(\mathbf{x}_{T+1}) \\
\mathbf{x}_{T+2} = [\hat{y}_{T+1}, y_t, \ldots, y_{T+2-\tau}]^\top \quad &\rightarrow \quad y_{T+2} \sim \mathcal{N}(m(\mathbf{x}_{T+2}), v(\mathbf{x}_{T+2})) \\
&\qquad \hat{y}_{T+2} = m(\mathbf{x}_{T+2}) \\
&\vdots \\
\mathbf{x}_{T+k} = [\hat{y}_{T+k-1}, \hat{y}_{T+k-2}, \ldots, \hat{y}_{T+k-\tau}]^\top \quad &\rightarrow \quad y_{T+k} \sim \mathcal{N}(m(\mathbf{x}_{T+k}), v(\mathbf{x}_{T+k})) \\
&\qquad \hat{y}_{T+k} = m(\mathbf{x}_{T+k})
\end{aligned}
$$

This setup does not account for the uncertainty induced by each successive prediction. For each recursion, the current state vector is considered deterministic, ignoring the fact that the previous predictions that it contains as elements are in fact random variables distributed according to the predictive distribution given by the model. The predictive variance obtained after a few recursive predictions is therefore way too small, and completely fails to reflect the accumulated uncertainty.

4.2.2 Propagating the Uncertainty

Using the results derived in Sect. 4.1, we propose to formally incorporate the uncertainty information about each successive iterated prediction. That is, as we predict ahead in time, we now view the lagged outputs as random variables. The input vectors, will therefore be random variables as they incorporate predictions recursively, with multivariate Gaussian distributions $\mathbf{x}_t \sim \mathcal{N}(\mathbf{u}_t, \mathbf{S}_t)$. Suppose as before, that data samples have been observed up to time T, and we wish to predict k steps ahead. Let us see the step by step evolution of the input and output distributions:

- at $t = T+1$, $\mathbf{x}_{T+1}$ is deterministic:

  $$\mathbf{u}_{T+1} = \begin{bmatrix} y_T \\ \ldots \\ y_{T+1-\tau} \end{bmatrix} \quad \text{and} \quad \mathbf{S}_{T+1} = \begin{bmatrix} 0 & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 0 & \ldots & 0 \end{bmatrix} ,$$

  and since $\mathbf{x}_{T+1} = \mathbf{u}_{T+1}$, $\mathbf{y}_{T+1}$ has a Gaussian predictive distribution given by:

  $$p(y_{T+1}|\mathbf{x}_{T+1}) \sim \mathcal{N}(m(\mathbf{u}_{T+1}), v(\mathbf{u}_{T+1})) ,$$

- at $t = T + 2$, $\mathbf{x}_{T+2}$ is random, with one normally distributed component:

$$\mathbf{u}_{T+2} = \begin{bmatrix} m(\mathbf{u}_{T+1}) \\ \dots \\ y_{T+2-\tau} \end{bmatrix} \quad \text{and} \quad \mathbf{S}_{T+2} = \begin{bmatrix} v(\mathbf{u}_{T+1}) & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & 0 \end{bmatrix} ,$$

  and we know from Sect. 4.1 that the marginal predictive distribution of $\mathbf{y}_{T+2}$ is generally not Gaussian anymore. We know however how to approximate it by a Gaussian:

$$p(y_{T+2}|\mathbf{u}_{T+2}, \mathbf{S}_{T+2}) \sim \mathcal{N}(m(\mathbf{u}_{T+2}, \mathbf{S}_{T+2}), v(\mathbf{u}_{T+2}, \mathbf{S}_{T+2})) \ .$$

  We will in a similar way repeatedly approximate distributions of the next iterated predictions by Gaussians. This does also ensure that the inputs remain Gaussian, which is necessary for us to be able to compute the marginal predictive distributions.

- at $t = T + k$, supposing that $k > \tau$, $\mathbf{x}_{T+k}$ is Gaussian with full covariance matrix:

$$\mathbf{u}_{T+k} = \begin{bmatrix} m(\mathbf{u}_{T+k-1}, \mathbf{S}_{T+k-1}) \\ \dots \\ m(\mathbf{u}_{T+k-\tau}, \mathbf{S}_{T+k-\tau}) \end{bmatrix} \quad \text{and}$$

$$\mathbf{S}_{T+k} = \begin{bmatrix} v(\mathbf{u}_{T+k-1}, \mathbf{S}_{T+k-1}) & \mathrm{cov}(y_{T+k-1}, y_{T+k-2}) & \dots & \mathrm{cov}(y_{T+k-1}, y_{T+k-\tau}) \\ \mathrm{cov}(y_{T+k-1}, y_{T+k-2}) & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \mathrm{cov}(y_{T+k-1}, y_{T+k-\tau}) & \dots & \dots & v(\mathbf{u}_{T+k-\tau}, \mathbf{S}_{T+k-\tau}) \end{bmatrix} ,$$

  where a new term, the covariance between pairs of probabilistic predictions, i.e. $\mathrm{cov}(y_{T+k-1}, y_{T+k-2})$, has entered the computation of the covariance of the inputs. We discuss in Sect. 4.2.3 how to compute them. We finally reach our objective, the marginal predictive distribution of $\mathbf{y}_{T+k}$, which we have obtained after k recursive Gaussian approximations:

$$p(y_{T+k}|\mathbf{x}_{T+k}) \sim \mathcal{N}(m(\mathbf{u}_{t+k}, \mathbf{S}_{t+k}), v(\mathbf{u}_{t+k}, \mathbf{S}_{t+k})) \ .$$

4.2.3 Input distribution

We can easily find a general expression for the Gaussian approximation to the input distribution. At time $t = T + k + 1$, the covariance matrix $\mathbf{S}_{T+k+1}$ of the state is computed by removing its last row and column, and inserting a new first row and column, which are the transpose of the other, since the covariance matrix is symmetric. This new first column is given by:

$$[\mathbf{S}_{T+k+1}]_{1:\tau,1} = \begin{bmatrix} v(\mathbf{u}_{T+k}, \mathbf{S}_{T+k}) \\ \mathrm{cov}(y_{T+k}, y_{T+k-1}) \\ \cdots \\ \mathrm{cov}(y_{T+k}, y_{T+k-\tau+1}) \end{bmatrix} = \begin{bmatrix} v(\mathbf{u}_{T+k}, \mathbf{S}_{T+k}) \\ \mathrm{cov}(y_{T+k}, \tilde{\mathbf{x}}_{T+k}) \end{bmatrix} \quad (4.20)$$

where $\tilde{\mathbf{x}}_{T+k}$ is a shorter version of the state vector $\mathbf{x}_{T+k}$ where the last element has been truncated: as we incorporate the new prediction in the state vector, we need to get rid of the oldest prediction. For simplicity in the notation, we will compute the covariance $\mathrm{cov}(y_{T+k}, \mathbf{x}_{T+k})$ and then throw away the last element of that vector to obtain $\mathrm{cov}(y_{T+k}, \tilde{\mathbf{x}}_{T+k})$. We have

$$\begin{aligned} \mathrm{cov}(y_{T+k}, \mathbf{x}_{T+k}) &= \mathrm{E}_{\mathbf{x}_{T+k}}[\mathrm{E}_{y_{T+k}}[y_{T+k}\, \mathbf{x}_{T+k}]] - \mathrm{E}_{\mathbf{x}_{T+k}}[\mathbf{x}_{T+k}]\ \mathrm{E}_{y_{T+k}}[y_{T+k}] \ , \\ &= \mathrm{E}_{\mathbf{x}_{T+k}}[m(\mathbf{x}_{T+k})\, \mathbf{x}_{T+k}] - m(\mathbf{u}_{T+k}, \mathbf{S}_{T+k})\, \mathbf{u}_{T+k} \ , \end{aligned} \quad (4.21)$$

where we have rewritten the joint distribution as the input distribution times the predictive: $p(y_{T+k}\, \mathbf{x}_{T+k}) = p(y_{T+k}|\mathbf{x}_{T+k})\, p(\mathbf{x}_{T+k})$, and $\mathrm{E}_{\mathbf{x}_{T+k}}$ denotes the expectation over $p(\mathbf{x}_{T+k})$, and $\mathrm{E}_{y_{T+k}}$ that over $p(y_{T+k}|\mathbf{x}_{T+k})$. We now need to compute the term:

$$\begin{aligned} \mathrm{E}_{\mathbf{x}_{T+k}}[m(\mathbf{x}_{T+k})\, \mathbf{x}_{T+k}] &= \int \mathbf{x}_{T+k}\, m(\mathbf{x}_{T+k})\, p(\mathbf{x}_{T+k})\, \mathrm{d}\mathbf{x}_{T+k} \ , \\ &= \sum_{i=1}^{T} \beta_i \int \mathbf{x}_{T+k}\, K(\mathbf{x}_{T+k}, \mathbf{x}_i)\, p(\mathbf{x}_{T+k})\, \mathrm{d}\mathbf{x}_{T+k} \ . \end{aligned} \quad (4.22)$$

We know from the computation of the predictive mean, given by (4.8), that $K(\mathbf{x}_{T+k}, \mathbf{x}_i)\, p(\mathbf{x}_{T+k})$ is Gaussian in $\mathbf{x}_{T+k}$, with normalising constant l_i given by (4.10).[4] All that remains for us to do is to compute the mean of that Gaussian distribution, which using the algebra in Sect. A.2, is seen to be given by:

$$\mathbf{c}_i = \mathbf{S}_{T+k}(\mathbf{S}_{T+k} + \boldsymbol{\Theta})^{-1}\mathbf{x}_i + [\mathbf{I} - \mathbf{S}_{T+k}(\mathbf{S}_{T+k} + \boldsymbol{\Theta})^{-1}]\mathbf{x}_{T+k} \ , \quad (4.23)$$

and is a convex combination of $\mathbf{x}_i$ and $\mathbf{x}_{T+k}$. We now can express (4.22) in a simpler form:

$$\mathrm{E}_{\mathbf{x}_{T+k}}[m(\mathbf{x}_{T+k})\, \mathbf{x}_{T+k}] = \sum_{i=1}^{T} \beta_i\, l_i\, \mathbf{c}_i \ , \quad (4.24)$$

and reach our goal, the covariance between y_{T+k} and $\mathbf{x}_{T+k}$:

$$\mathrm{cov}(y_{T+k}, \mathbf{x}_{T+k}) = \mathbf{S}_{T+k}(\mathbf{S}_{T+k} + \boldsymbol{\Theta})^{-1} \sum_{i=1}^{T} \beta_i l_i (\mathbf{x}_i - \mathbf{u}_{T+k}) \ . \quad (4.25)$$

[4] Where we simply need to set $\mathbf{S} = \mathbf{S}_{T+k}$ and $\mathbf{u} = \mathbf{u}_{T+k}$.

Not surprisingly, the larger the lengthscales of the model, the more smoothness and the smaller the covariance between predictions. Also, for small input uncertainties ($\mathbf{S}_{T+k}$ small), the covariances will also be small. This is due to the fact that a part of the uncertainty of y_{T+k} stems from the uncertainty of $\mathbf{x}_{T+k}$. If $\mathbf{x}_{T+k}$ was certain, its covariance with $\mathbf{y}_{T+k}$ would obviously be zero.

4.2.4 Experiments

Is this section we report the experiments we performed in (Quiñonero-Candela et al., 2003b), where the goal was to compare the performance of two methods for propagating the uncertainty in iterated time-series predictions. The first method is based on approximately computing the first two moments of the marginal predictive distribution by using Taylor expansions, and was derived by Girard et al. (2003), and the second method is based on the exact computation of mean and variance, that we describe in Sect. 4.1.1. Both methods are based on a recursive Gaussian predictive density (RGPD) computation, as described in Sect. 4.2.2. For convenience we will give names to both methods: the method based on approximately computing the moments will be called “approximate-RGPD”, and the method based on exact computation of the moments will be called “exact-RGPD”.

We will use the same dataset as in Sect. 2.3, the Mackey-Glass chaotic time series (Mackey and Glass, 1977), which is well-known for its strong non-linearity. In Sect. 2.3 we described the dataset, and we showed that non-linear models, in particular RVMs, have a prediction error four orders of magnitude lower than optimised linear models. The inputs are formed by $\tau = 16$ samples. We train a GP model with squared exponential covariance function, (3.1), on only 100 examples — enough to obtain a 1-step ahead normalised mean squared error on the order of 10^{-4}. Besides, we normalise the data and contaminate it with a small amount of Gaussian noise with variance 10^{-3}. Figure 4.2 shows the result of making 100 iterative predictions using a GP model, both for the exact-RGPD and the approximate-RGPD methods. By informal visual inspection, the error-bars of the exact-RGPD seem to be better than those of the approximate-RGPD. Consequently the exact-RGPD produces a better predictive density, which we show in Fig. 4.3. The mean value of the predictions seems also to be a slightly closer to the true target values for the exact-RGPD than for the approximate-RGPD.

In order to better evaluate the performance of the proposed methods, for a given prediction horizon, we compute the negative log predictive density, the squared error and the absolute error. While the two last measures only take into consideration the mean of the Gaussian predictive distribution, the first one also

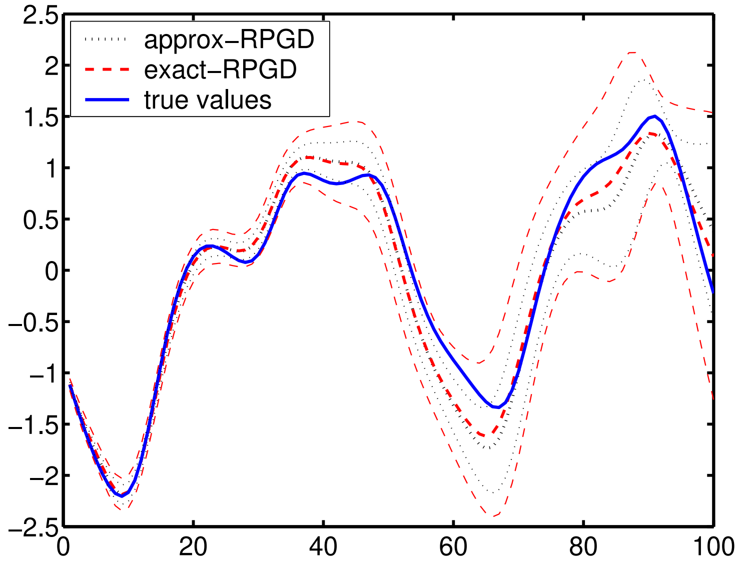

Figure 4.2: 100 iterated predictions for the exact-RGPD (dashed) and approximate-RGPD (dotted): for each the thicker lines represent the mean of the predictive distributions and the two thinner lines around represent the upper and lower bounds of the 95% confidence interval of the Gaussian predictive distributions. The solid line shows the true target values.

takes into account its variance. We average over 200 repetitions with different starting points (chosen at random from the series), and represent averages of the three loss measures for prediction horizons ranging from 1 to 100. Figure 4.3 shows the results. The means are slightly better for the exact-RGPD, but the predictive distribution is much improved. The better error-bars obtained by the exact-RGPD result in a lower value of the negative log predictive density for all values of the prediction horizon. The performance of the naïve iterative method is identical to that of the approximate-RGPD in terms of absolute and squared error. In terms of predictive density (since it produces unrealistic small error-bars) its performance is so poor that it is not worth reporting.

4.2.5 Conclusion

In Sect. 4.1 we have derived analytical expressions for the exact computation of the mean and variance of the marginalised predictive distribution for uncertain

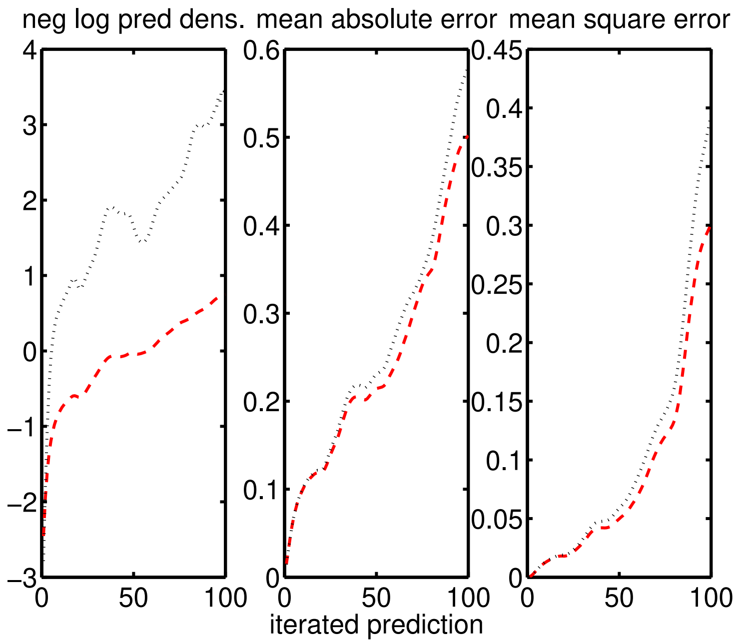

Figure 4.3: Negative log predictive density, mean absolute error and mean squared error as a function of the iterative prediction horizon for the exact-RGPD method (dashed) and for the approximate-RGPD (dotted). Averages over 200 repetitions.

Gaussian test inputs. This analytical expressions are valid for GPs and RVMs (extended linear models) with Gaussian or polynomial covariance or basis functions. Our results extend the approximate method presented in (Girard et al., 2003), where the mean prediction was unaffected by the input uncertainty. In our case the input uncertainty biases the mean prediction, by smoothing, which is interesting in itself for predictions on noisy inputs. Furthermore, in the context of iterated time-series forecasting, described in Sect. 4.2.2, our exact-RGPD not only gives much better error-bars, but the mean predictions are closer to the true values, both in terms of absolute and squared error. Improving the mean predictions was however not our primary objective, and we are essentially satisfied with having obtained sensible predictive variances. Our expressions have also recently been applied to reinforcement learning (Rasmussen and Kuss, 2004).

4.3 On Learning GPs with Uncertain Inputs

We have touched the issue of using noisy inputs with GPs (and RVMs). In Sect. 4.1 we have shown how to make *predictions* with a previously trained GP (or RVM) on an uncertain test point. But this begs the obvious question: how do we *train* GPs[5] using noisy inputs?

When a regression or classification task is to be solved by training a statistical model on noisy and/or missing inputs, the quality of the model we estimate may suffer if we do not correctly account for the uncertainty inherent in the training set. (This effect may actually occur in two separate but related ways, one due to training with noisy inputs and the other due to extra noise in the outputs caused by the noise in the inputs.) Statisticians have investigated this problem in several guises: "total least-squares" (Golub and Loan, 1980) modifies the cost of a regression problem to encourage the regressor to pass near both noisy targets and uncertain inputs; the "errors-in-variables" model (Dellaportas and Stephens, 1995; Carroll et al., 1995) deals directly with noisy inputs, both in their natural form and as a method of fitting complex nonlinear models by creating correlated virtual variables that thus have correlated noises. "Multiple-imputation" (Rubin, 1987) combines the outputs of a series of models trained with different replacements of the missing data. Recent work in machine learning has also addressed this problem, either by attempting to integrate over missing inputs by learning the entire input distribution (Ghahramani and Jordan, 1994), or integrating over specific noisy points using approximate distributions estimated during training (Tresp et al., 1994).

Let us formulate the problem for GPs, the standard training of which we described in Sect. 3.1. In Sect. 4.1 we considered the case where a test input was random, and we only had access to its normal distribution $\mathbf{x}_* \sim \mathcal{N}(\mathbf{u}, \mathbf{S})$. Here we consider that we do not have access to the training inputs $\{\mathbf{x}_i\}$, but to a noisy version of them:

$$\mathbf{u}_i = \mathbf{x}_i + \nu_i \ , \tag{4.26}$$

where the random variable ν_i represents i.i.d. Gaussian noise. The inputs are assumed to be independent given their statistics, each input $\mathbf{x}_i$ is a stochastic variable with distribution $\mathcal{N}(\mathbf{u}_i, \mathbf{S})$. This assumption is reasonable provided that the input distribution varies very slowly compared to the standard deviation of the input noise.[6] If we let the inputs be organised as rows in matrix X, the input distribution, that we will also refer to as the *prior* distribution on the

[5]RVMs being a special instance of GPs, and since we are more interested in the latter, we will not explicitly consider the particular case of RVMs.

[6]The author is grateful to Chris Williams for a comment on this.

inputs, can be written as:

$$p(X|\Phi) = \prod_{i=1}^{N} p(\mathbf{x}_i|\phi_i) \ . \tag{4.27}$$

We assume that we observe the statistics $\phi_i = \{\mathbf{u}_i, \mathbf{S}_i\}$ of each input $\mathbf{x}_i$, and we write $\Phi = \{\phi_i\}$ as the set of statistics of the distributions of all inputs. Without loss of generality and for simplicity, we will only consider the case where for a given input the noise is isotropic, that is that $\mathbf{S}_i = s_i\,\mathbf{I}$. We are also given a corresponding set of training targets, $\mathbf{y} = [y_1, \ldots, y_N]^\top$, that differ from the function values $f(\mathbf{x}_i)$ by Gaussian iid. noise of variance σ^2. If we are now given a new certain input $\mathbf{x}_*$ and asked to provide with the predictive distribution of its associated target y_*, the Bayesian dream would be to integrate over uncertain training inputs and over the hyperparameters of the covariance function, by defining a prior $p(\theta)$ over the latter:[7][8]

$$p(y_*|\mathbf{x}_*, \mathbf{y}, \Phi) = \frac{1}{p(\mathbf{y}|\Phi)} \iint p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \theta)\, p(\mathbf{y}|X, \theta)\, p(X|\Phi)\, p(\theta)\, \mathrm{d}X \mathrm{d}\theta \ , \tag{4.28}$$

but this is unfortunately analytically intractable for most covariance functions, and in particular for the squared exponential, (3.1), that we consider here. There are three options we could consider. The first and most obvious one is approximate the integration by sampling both over the inputs and the hyperparameters, using Markov Chain Mote-Carlo (MCMC) sampling methods. There are two other approaches which we will discuss here. One is to do MAP on the hyperparameters, and sample over the weights. To see how to do this let us re-write the troublesome integral using Bayes rule:

$$\begin{aligned} p(y_*|\mathbf{x}_*, \mathbf{y}, \Phi) &= \iint p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \theta)\, p(\mathbf{y}|X, \theta)\, p(X|\Phi)\, \frac{p(\theta|\mathbf{y}, \Phi)}{p(\mathbf{y}|\theta, \Phi)}\, \mathrm{d}X \mathrm{d}\theta \\ &\approx \iint p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \theta)\, p(\mathbf{y}|X, \theta)\, p(X|\Phi)\, \frac{\delta(\theta_{MAP})}{p(\mathbf{y}|\theta, \Phi)}\, \mathrm{d}X \mathrm{d}\theta \\ &= \int p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{y}, X, \theta_{MAP})\, p(X|\mathbf{y}, \theta_{MAP}, \Phi)\, \mathrm{d}X \ , \end{aligned} \tag{4.29}$$

where we still need to sample from the posterior distribution over the uncertain inputs, $p(X|\mathbf{y}, \theta_{MAP}, \Phi)$. We also need to obtain the maximum of the posterior over the hyperparameters: $p(\theta|\mathbf{y}, \Phi) \propto p(\mathbf{y}|\theta, \Phi)\, p(\theta)$. We see that if we define an

[7]In Sect. 3.1 we had learned θ using Maximum Likelihood II (MLII), which is equivalent to the MAP solution with an flat prior.

[8]For notational convenience, we absorb the estimate of the output noise, σ^2, in the set of hyperparameters θ.

improper flat prior over θ, maximising the posterior is equivalent to maximising the marginal likelihood:

$$p(\mathbf{y}|\theta,\Phi) = \int p(\mathbf{y}|\theta,X)\, p(X|\Phi)\, \mathrm{d}X \ , \tag{4.30}$$

averaged over the input distribution. We will see in Sect.4.3.2 that this can easily be done with an EM algorithm where the inputs are treated as hidden variables.

In Sect. 4.3.1 we present a second approximation to (4.28), consisting in estimating both the hyperparameters and the uncertain inputs by doing MAP. To see what we mean let us write again the intractable integral, (4.28), using Bayes rule:

$$\begin{aligned} p(y_*|\mathbf{x}_*,\mathbf{y},\Phi) &= \iint p(\mathbf{y}_*|\mathbf{x}_*,\mathbf{y},X,\theta)\, p(\mathbf{y}|X,\theta)\, \frac{p(X,\theta|\mathbf{y},\Phi)}{p(\mathbf{y}|X,\theta)}\, \mathrm{d}X\mathrm{d}\theta \\ &\approx \iint p(\mathbf{y}_*|\mathbf{x}_*,\mathbf{y},X,\theta)\, p(\mathbf{y}|X,\theta)\, \frac{\delta(X_{MAP},\theta_{MAP})}{p(\mathbf{y}|X,\theta)}\, \mathrm{d}X\mathrm{d}\theta \\ &= p(\mathbf{y}_*|\mathbf{x}_*,\mathbf{y},X_{MAP},\theta_{MAP}) \ , \end{aligned} \tag{4.31}$$

where the difficulty lies now only the maximisation of the joint posterior over θ and X. Using again a flat prior over hyperparameters, the joint posterior is given by:

$$p(X,\theta|\mathbf{y},\Phi) \propto p(\mathbf{y}|X,\theta)\, p(X|\Phi) \ , \tag{4.32}$$

which can be seen as a form of penalised likelihood, where both inputs and hyperparameters are learned. This approach is in fact similar to a line of attack, pioneered by Weigend et al. (1996), in which rather than integrating over uncertain data, we try to train our model accounting for the noise in the inputs and at the same time use our model to help us infer their true values. This has the advantage of being much more tractable than the goal of integration, as well as producing as an output of our algorithm, "cleaned" versions of the noisy inputs which may be of interest in its own right. We will see that the intuition behind this first approach is the same as Weigend's: we set up a model that is allowed to move inputs away from their observed values if it substantially improves the fitting of the outputs, but not if they move too far. However, unlike Weigend's more heuristic cost function, our method is somewhat more founded, since it attempts to approximate a fully probabilistic model. One consequence of the MAP approach is that it avoids fitting over-smooth models and does not "underestimate the slopes" (Carroll et al., 1995). Another consequence is that we return a "cleaned" version of the input data, which can be useful in its own right. Of course this corrected data is only as reliable as our belief that the true underlying function is close to one of the family of functions our GP prior can generate. As we will discuss, this approach can easily lend itself to over-fitting, and we are forced to use a simulated annealing procedure to avoid the numerous undesirable spurious maxima.

4.3.1 Maximum Posterior Input Imputation

We want to maximise the joint posterior of hyperparameters and uncertain inputs, given by (4.32). We decide to minimise equivalently its negative logarithm:

$$\begin{aligned}\mathcal{P} &= -\log p(\mathbf{y}|X,\theta) - \sum_{n=1}^{N} \log p(\mathbf{x}_n|\phi_n) \\ &= N\,\log 2\pi + \frac{1}{2}\log|\mathbf{Q}| + \frac{1}{2}\mathbf{y}^\top \mathbf{Q}^{-1}\,\mathbf{y} + \frac{1}{2}\sum_{n=1}^{N}\log s_n + \frac{1}{2}\sum_{n=1}^{N}\frac{||\mathbf{x}_n - \mathbf{u}_n||^2}{s_n}\quad,\end{aligned} \tag{4.33}$$

where for convenience we have defined $\mathbf{Q} = \mathbf{K} + \sigma^2\,\mathbf{I}$. The task that remains is now to compute the gradient of $\mathcal{P}$ wrt. X and θ, and use some gradient ascent algorithm to perform the minimisation. Computing the gradient wrt. θ is a task that we already have done:

$$\frac{\partial\mathcal{P}}{\partial\theta_i} = -\frac{\partial}{\partial\theta_i}\log p(\mathbf{y}|X,\theta)\ ,$$

since it corresponds to the derivatives of the standard GP negative log evidence with known inputs, and we refer to (3.8) for the details. To compute derivatives wrt. the inputs, let us denote as usual the d-th component of vectors $\mathbf{x}_i$ and $\mathbf{u}_i$ by X_{id} and U_{id} respectively. We have:

$$\frac{\partial\,\mathcal{P}}{\partial X_{id}} = \frac{1}{2}\,\mathrm{Tr}(\mathbf{Q}^{-1}\frac{\partial\,\mathbf{K}}{\partial X_{id}}) - \frac{1}{2}\mathbf{y}^\top\mathbf{Q}^{-1}\frac{\partial\,\mathbf{K}}{\partial X_{id}}\mathbf{Q}^{-1}\mathbf{y} + \frac{X_{id}-U_{id}}{s_i}, \tag{4.34}$$

and the gradient of $\mathbf{K}$ with respect to X_{id} is given by

$$\left[\frac{\partial\mathbf{K}}{\partial X_{id}}\right]_{lm} = \begin{cases} 0 & l,m \neq i \\ -\frac{1}{\theta_d^2}\mathbf{K}_{lm}(X_{ld}-X_{md}) & \text{otherwise}\end{cases}. \tag{4.35}$$

We obtain two things from the minimisation of $\mathcal{P}$ with respect to θ and X: the parameters of the Gaussian Process, and an estimate of the true location of the inputs of the training data. The model is inferred from the estimated data, and the estimated data are inferred both from the prior distributions on the inputs and from the estimated GP model.

4.3.1.1 An Ideal Example

For convenience, we will refer to our algorithm for training a Gaussian Process while simultaneously estimating the true locations of the inputs (or *cleaning* them) as “cleanGP”. In Fig. 4.4 we illustrate the way our algorithm works. We generate 20 training points uniformly spaced between -3 and 3. The corresponding outputs are given by $f(\mathbf{x})_i = \sin(x_i)$ and contaminated with a small amount of Gaussian i.i.d. noise, of standard deviation 10^{-2}. These are the “clean” training points (blue circles); none of the models actually sees this data. We now add Gaussian i.i.d. noise to the inputs of the clean training set, of standard deviation 1.2 (large, about three times the space between two consecutive inputs, so that occasionally inputs will be “swapped”) and obtain the green crosses, which are the “noisy inputs training data”. We train a standard Gaussian Process on the noisy inputs data and make predictions on the clean inputs (mean shown by a green dashed line). We now train our cleanGP on the same noisy data (green crosses). The red squares in the figure show the “cleaned” data, i.e. the most likely location under our cleanGP model. (A few points show arrows between crosses and corresponding squares.) Finally, we also make predictions with the resulting cleanGP on the original grid of equally spaced clean inputs (mean shown by a red solid line). The cleanGP has achieved our two goals. One the one hand it fits a better model (for example, it avoids underestimating the slopes). On the other hand, it produces an estimate of the “true” location of each noisy inputs in the training data.

An crucial subtle confession has to be made at this point: as we detail in section 4.3.1.2, to avoid over-fitting the training is done with knowledge of the level of output noise.

4.3.1.2 Spurious Global Minima in Training

Training GPs with uncertain inputs is unfortunately not as simple as the maximisation of the joint posterior over uncertain inputs and hyperparameters that we have just proposed. In practice, when we pass the expression of the negative log posterior and its derivatives to a gradient (or conjugate gradient) descent algorithm, the model quickly gets driven to one of very many undesirable spurious global minima with extremely poor generalisation performance. This behaviour can be understood by examining the two terms of $\mathcal{P}$. The first term is the negative log likelihood of the GP given the estimated inputs X; this term can increase indefinitely if the estimated inputs are placed so that the GP can exactly fit the resulting training data, shrinking the estimated output noise variance virtually to zero. The second term of $\mathcal{P}$ is the negative log prior distribution on the

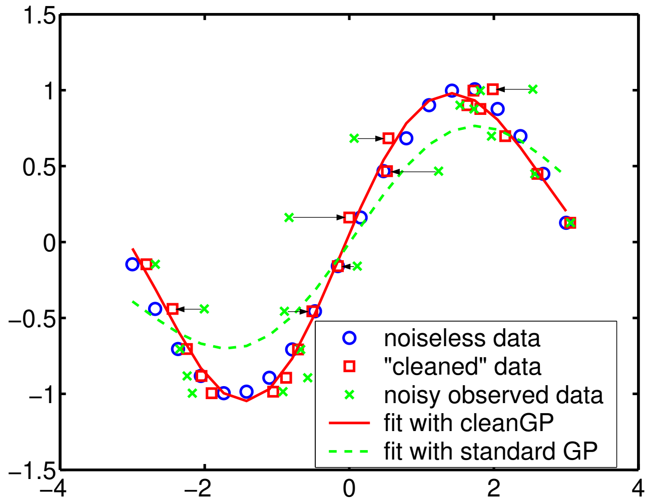

Figure 4.4: Illustration of how the data are cleaned. The blue circles are the training data with noiseless inputs. Noise is added to the inputs to produce the green crosses, which the "input noisy" training data presented to the Gaussian Process. A standard GP that does not clean the inputs produces an over-smooth model (green dashed curve). Our algorithm allows the GP to clean the input data. The red boxes show the estimates of the locations of the "true" inputs. We have put some arrows in the figure to show where the model has moved the noisy points. The red solid line shows the curve fitted to the data by the cleanGP. (The regression curves for both models are the mean of the predictive distribution where the test inputs are a uniform noiseless grid) An important detail: we fixed the output noise to its actual value instead of learning it.

inputs. Although this term penalises estimates of the inputs that lie far away from the observed inputs (means of the priors), it is ultimately no match for the gain in the GP log likelihood term, unless the input noise is essentially zero. In short: the GP model desperately tries to attract the inputs to a location where it can perfectly fit the training data. As one would expect, cleanGP over-fits in an extreme manner.

Random restarts of the optimisation (with small perturbations of the initialisation) also show that the final solution is quite variable, even for fixed training data. Following an EM-like approach, where the GP parameters were optimised holding the estimated inputs fixed, and then the positions of the inputs were optimised fixing the parameters (and so on), exhibited over-fitting far more slowly, and thus gave us the intuition that the optimisation process was passing near a good solution.

One solution we have found to the over-fitting problem is to use an annealing type of training, where we fix the output noise level to some high value and optimise the rest of the parameters (i.e. lengthscales of the GP covariance in θ and the estimated inputs X). We then lower the value of the output noise, and retrain the rest of the parameters, starting from their previous estimates. In Fig. 4.5 we illustrate this process, for the "sinc" toy data: we generate 50 uniformly spaced inputs points between -10 and 10. We build the corresponding outputs as $y_i = sin(x_i)/x_i$ plus a small amount of Gaussian i.i.d. noise, of standard deviation 10^{-2}. This is the clean training data. To generate the noisy inputs training data we add noise to the inputs, of standard deviation 1. We build a test set in a similar manner, generating 100 inputs uniformly at random between -10 and 10, and computing the corresponding outputs. The test negative log predictive density (or test energy) is represented against the output noise variance for a given solution. The blue line with dots contains the trajectory of the solutions obtained when training cleanGP with annealing. The diamond shows the optimal solution. The red circles show the solutions obtained when initialising cleanGP at the given level of output noise, keeping that noise level fixed, and optimising for 20 random initialisations of the remaining GP parameters. It can be seen that several different solutions are attained when the noise level diminishes, and that they all generalise very poorly. The green star is the solution obtained when training a standard Gaussian Process, allowing for optimisation of all the parameters. We plot its estimate of the output noise versus its test error. Of course, knowing when to stop the annealing process is difficult in practice. We have had limited success with monitoring the magnitude of the gradient with respect to the output noise level, and stopping the annealing procedure at the minimum of this quantity.

It is not satisfactory to have to know in advance the variance of the output noise in order to be able to use the cleanGP approach. It would be much preferable to be able to estimate it. One would hope to be able to improve on the noise estimates from a standard GP, that in the presence of noise in the inputs would tend to over-estimate the output noise.

4.3.2 A Stochastic EM Approach

Let us now address the alternative approach to learning GPs with uncertain inputs, based on MAP estimates of the hyperparameters θ, but integration over the uncertain inputs X. We explained previously that the difficulty is centred on maximising the posterior over θ marginalised over the inputs. This maximisation we said, is equivalent to maximising the marginal likelihood, (4.30), averaged

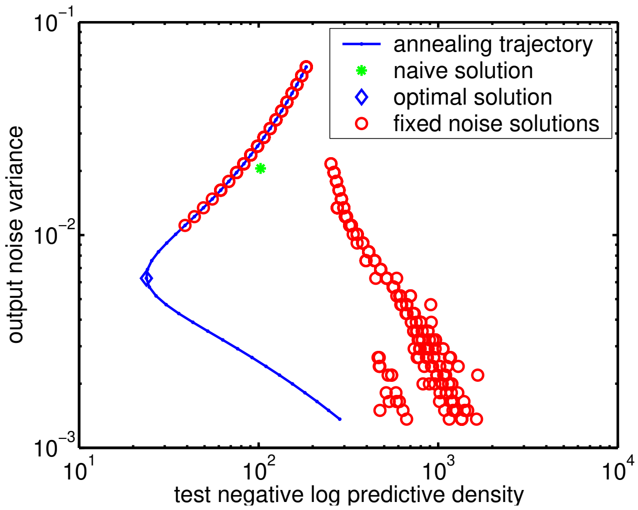

Figure 4.5: The blue line with dots represents the annealing trajectory for the cleanGP. The green star represents the optimal solution obtained for a standard GP. The red circles represent the solutions obtained for 20 repetitions of training of a cleanGP with fixed output noise and a random initialisation of the remaining parameters of the GP.

over the uncertain inputs, that we will call "marginal evidence". Let us set this maximisation up in an EM framework. The log marginal evidence is given by:

$$\begin{aligned} \mathcal{L} = \log p(\mathbf{y}|\theta, \Phi) &= \int p(\mathbf{y}, X|\theta, \Phi) \, \mathrm{d}X \\ &= \log \int q(X) \, \frac{p(\mathbf{y}, X|\theta, \Phi)}{q(X)} \, \mathrm{d}X \\ &\geq \int q(X) \, \log \frac{p(\mathbf{y}, X|\theta, \Phi)}{q(X)} \, \mathrm{d}X = \mathcal{F}(q, \theta) \ , \end{aligned} \tag{4.36}$$

where $q(X)$ is an arbitrary distribution of X and we have used Jensen's inequality (with the concavity of the logarithm) to produce a lower $\mathcal{F}(q, \theta)$ to $\mathcal{L}$. We described EM learning in Sect. 2.2.1, and we refer the reader to that section for the details. We basically need to iteratively estimate the distribution $q(X)$ for fixed θ, the E-step, and to estimate θ for a fixed $q(X)$.

The E-step corresponds to minimising the Kullback-Leibler divergence between $q(X)$ and the posterior on the inputs. We choose to perform a "stochastic" E-step, where instead of giving a parametric form to $q(X)$, we directly sample from the posterior distribution of X. Such posterior is given by:

$$p(X|\mathbf{y}, \theta, \Phi) \propto p(\mathbf{y}|X, \theta)\, p(X|\Phi) \ , \tag{4.37}$$

and we notice that it is proportional to the penalised marginal likelihood for which we just computed the derivatives wrt. X in Sect. 4.3.1. It is therefore straightforward to sample from this posterior using for instance Hybrid Monte-Carlo (see for example Neal, 1993), which is the MCMC method we choose to use for this task.

The M-step corresponds to maximising the average of the joint distribution $p(\mathbf{y}, X|\theta, \Phi)$ over $q(X)$. Suppose we have obtained a set of samples from the posterior $\{X_\tau | \tau = 1, \ldots, \mathrm{T}\}$. The approximate average over the posterior is given by:

$$\hat{\theta} = \arg\max_\theta \frac{1}{\mathrm{T}} \sum_{\tau=1}^{T} \log p(\mathbf{y}|X_\tau, \theta) \ , \tag{4.38}$$

where we have omitted the term $\log p(X|\Phi)$ because it does not depend on θ. The maximisation in the M-step is also straightforward for us, since for a given sample from the posterior, $\log p(\mathbf{y}|X_\tau, \theta)$ is simply the log evidence, and we explained in Sect. 3.1 how to compute it and its derivatives wrt. θ. The M-step implies simply an average of such log evidences and their derivatives.

To illustrate this learning procedure we generate a toy example, based on the toy "sinc" data described previously in Sect. 4.3.1.2. We train on the noisy input data with the stochastic EM until the likelihood seems to reach a stationary region. The results are displayed in Fig. 4.6. In the top panel we represent the noisy inputs training data with thick black crosses, and their noiseless input counterparts with thin circles (to give a visual impression of how much the inputs have been shifted). We take 1000 samples from the posterior on the uncertain inputs and plot them with gray dots. It is quite interesting to see, especially around the main lobe of the "sinc" function, how the samples seem to cover the "right" regions. At the same time the posterior seems to be broad enough to avoid over-fitting at the M-step. We now take the samples from the posterior and use them to approximate the intractable integral in (4.29), that gives us the predictive distribution. We train a standard GP on the noisy input data, treating the inputs as if they were noiseless, and compute its predictive distribution. We display the predictive distributions on the bottom panels: on the right for the standard GP, and on the left for the stochastic EM learning procedure. We see two things: the standard GP is too smooth, and it has larger predictive variances than the GP with stochastic EM. The actual standard deviation of the output noise is 0.05; the standard GP over-estimates it at 0.16, and the stochastic EM GP makes a quite accurate estimate at 0.06.

The computational cost of the stochastic EM is quite high. In the E-step, the covariance matrix needs to be inverted once for every leapfrog step in the Hybrid Monte-Carlo scheme, for each sample that is drawn. In the M-step, it needs to be inverted once for each sample from the posterior. This has the effect of multiplying the already high computational cost of training GPs. Additionally, the larger the training set and the dimension of the inputs, the more samples one should take in the E-step. In a nutshell, the stochastic EM approach is severely limited in practice by its high computational cost, to training sets of a few hundred cases.

4.3.3 Conclusion

We have discussed the issue of training GPs when the inputs are also contaminated with noise, of known Gaussian distribution. We have shown that it involves an analytically intractable integration. We have presented two preliminary approximations to solving this integral. The first one is based on a joint maximisation of the joint posterior on uncertain inputs and GP hyperparameters. We have shown that it suffers heavily from over-fitting, which is not surprising given the very large number of degrees of freedom. We have noticed that carefully controlling the value of the output noise, and reducing it in an annealing procedure allows to overcome the over-fitting problem. However, there does not seem to be any obvious stopping criterion for the annealing. We therefore need to know the actual output noise, which is a serious limitation. To overcome this limitation, we have proposed to sample from the posterior distribution on the inputs instead of optimising, and to still maximise with respect to the hyperparameters. The procedure can be naturally formulated as a stochastic EM algorithm, where in the E-step we sample from the posterior on the inputs, and in the M-step we maximise the average of the usual GP log evidence evaluated at the samples from the posterior. In preliminary toy experiments, the method seems promising: it quite well estimates the output noise, and seems to reach sensible posterior distributions on the inputs. Its practical use is however limited by its high computational cost. Future work will involve the use of variational methods, of the guise used in Sect. 2.2.2, to avoid having to use MCMC methods.

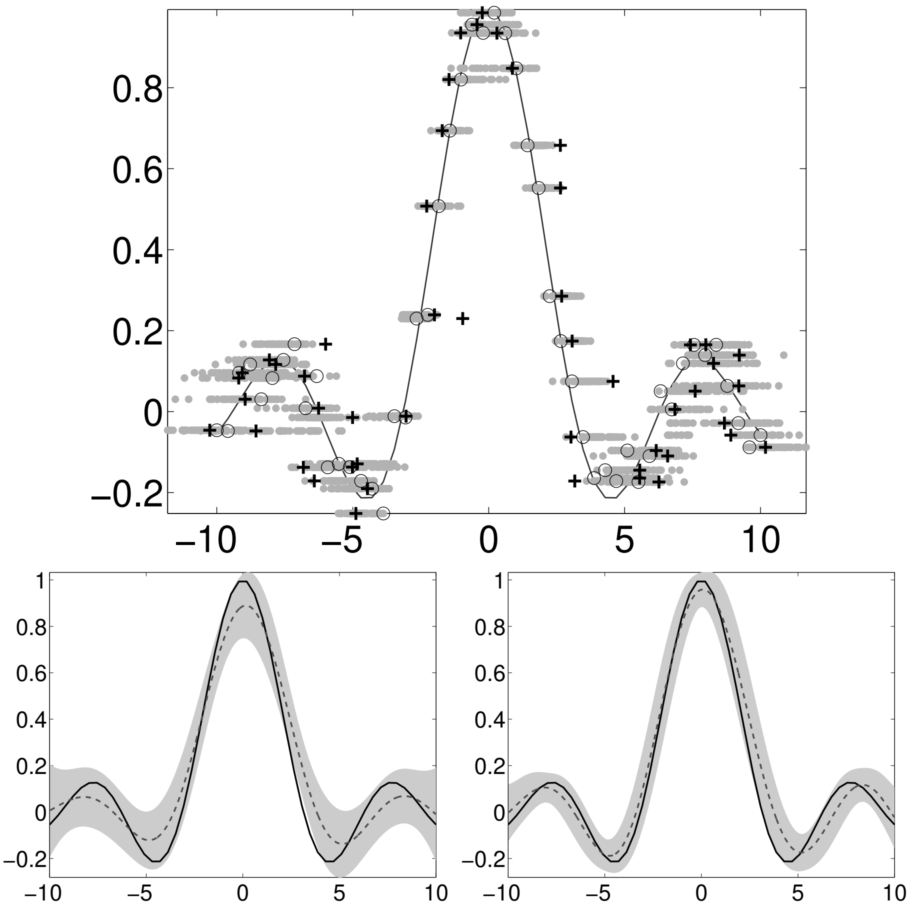

Figure 4.6: Stochastic EM for learning GPs on noisy inputs in action. *Top panel:* the circles show the training data with noiseless inputs, and the crosses that data after adding noise to the inputs. The models are trained on the crosses. The gray dots are samples from the posterior distribution over the uncertain inputs. *Bottom:* the dashed lines are the means, and the shaded areas represent the 95% confidence intervals of the predictive distributions for a standard GP (*left*) and for a GP trained with the stochastic EM (*right*). The standard GP ignores the randomness of the inputs, produces over-smooth predictions and over-estimates the output noise.

Chapter 5

Discussion

In this thesis we have first studied Relevance Vector Machines (RVMs). We have seen that sparseness does not arise from a Bayesian treatment, but rather from an approximation to it. We maintain the view that full Bayesian approaches do not yield sparse solutions, since it is hard to conceive a prior that would imply posteriors with exactly zero probability mass on regions where some of the model weights are non-zero. Approximate Bayesian treatments imply choosing "convenience" priors, that rather than expressing our genuine beliefs about how the functions should look like, are simply chosen because of analytical reasons, and because like for the RVM, they happen to enforce the desired sparseness. We have shown that for the RVM, and also for the Reduced Rank Gaussian Process (RRGP), these "convenience" priors correspond to counterintuitive assumptions about the functions. These in turn result in inappropriate predictive variances, that for instance are larger the closer to the training data, and smaller as we move away from it. We believe that one important justification for using a probabilistic, or a Bayesian framework is precisely to obtain probabilistic predictions. If the predictive distributions have inappropriate predictive variances, the probabilistic nature of predictions becomes useless. This fact has motivated us to slightly modify the posterior at prediction time, to include an additional basis function centred at the test inputs. This increases somewhat the computational cost of making predictions, but guarantees appropriate predictive variances, while being still much cheaper than the non-sparse model. For RVMs, we have also discussed the difficulty of simultaneously learning the parameters of the basis functions and the hyperparameters of the prior on the weights. It is interesting to note that this difficulty is shared by RRGPs, where selecting the support set and learning the covariance hyperparameters is challenging and may lead to over-fitting. This difficulty is in contrast on the one hand with learning GPs by maximising the evidence, where there is no sparsity, and the hyperparameters can be learned, and on the other hand with the RVM with fixed basis functions, where the lengthscales are fixed but a support set of inputs can be learned. It will be interesting to investigate frameworks that allow learning both support sets and parameters of the basis functions.

For the RVM, we have presented a very simple and computationally efficient incremental approach to training, the Subspace EM (SSEM) algorithm. Although we have mentioned our awareness of the existence of two other approaches, that of Tipping and Faul (2003), and that of D'Souza et al. (2004), we have not compared them to our method. It will be interesting to do so, from the theoretical but also from the practical point of view. The situation is similar for sparse GPs. We have mentioned a relatively extensive list of recent methods that have been proposed to increase the computational efficiency of GPs, but we have only compared our RRGP method based on the evidence to the method of Smola and Bartlett (2001). We are currently working on a theoretical survey of the proposed methods, with the intention of categorising the different sparseness paradigms. Future work will also include an extensive experimental comparison of the methods. Probably because the RVM is not primarily a sparse approximation to a GP, a thorough experimental comparison of both models seems to still be lacking in the literature. We have systematically found in our informal experiments, and in those presented in Sect. 2.5, that GPs were superior to RVMs. Future could involve a more exhaustive experimental comparison of both models.

We have successfully addressed the issue of predicting at an uncertain input with GPs and RVMs, and used this to propagate the uncertainty in recursive time-series predictions. This has allowed us to obtain sensible predictive variances when recursively predicting k-steps ahead, where naïve approaches are extremely over confident. Our propagation of uncertainty method has been successfully used by Rasmussen and Kuss (2004) in reinforcement learning. We have attempted to go beyond uncertainty at test inputs, and tried to solve the problem of training GPs on uncertain inputs. This has proven to be a much harder task. Our first approach consisted in imputing the "true" unseen inputs while learning the model hyperparameters. Though imputation might be interesting in its own right, as one would expect this setting can lead to extreme over-fitting if not used with much care. We had to resort to an annealing procedure of the output noise, which ultimately had to be known in advance. This is disappointing, and imposes a serious limitation on the usability of the method. The second approach we proposed gave promising results, allowing to learn the output noise and the covariance hyperparameters, while approximating the posterior distribution on the uncertain weights by sampling from it. Unfortunately having to sample dramatically increases the computational cost, limiting again the cases where the method can be used in practice. The results presented on the issue of training with uncertain inputs are nonetheless quite preliminary, and we feel that there is still a lot to be investigated. In particular, one should try approximating the posterior over uncertain inputs by means of variational approximations, to avoid having to sample. Conversely, one may want to study efficient sampling schemes, and sample not only over the inputs, but also over the hyperparameters.

APPENDIX A

Useful Algebra

A.1 Matrix Identities

The matrix inversion lemma, also known as the Woodbury, Sherman & Morrison formula states that:

$$(Z + UWV^\top)^{-1} = Z^{-1} - Z^{-1}U(W^{-1} + V^\top Z^{-1}U)^{-1}V^\top Z^{-1}, \tag{A.1}$$

assuming the relevant inverses all exist. Here Z is $n\times n$, W is $m\times m$ and U and V are both of size $n\times m$; consequently if Z^{-1} is known, and a low rank (ie. $m < n$) perturbation are made to Z as in left hand side of eq. (A.1), considerable speedup can be achieved. A similar equation exists for determinants:

$$|Z + UWV^\top| = |Z|\,|W|\,|W^{-1} + V^\top Z^{-1}U|\ . \tag{A.2}$$

Let the symmetric $n \times n$ matrix A and its inverse A^{-1} be partitioned into:

$$A = \begin{pmatrix} P & Q \\ Q^T & S \end{pmatrix}, \qquad A^{-1} = \begin{pmatrix} \tilde{P} & \tilde{Q} \\ \tilde{Q}^T & \tilde{S} \end{pmatrix}, \tag{A.3}$$

where P and $\tilde{P}$ are $n_1 \times n_1$ matrices and S and $\tilde{S}$ are $n_2 \times n_2$ matrices with

$n = n_1 + n_2$. The sub-matrices in A^{-1} are given in Press et al. (1992, p. 77):

$$\begin{aligned} \tilde{P} &= P^{-1} + P^{-1}QM^{-1}Q^T P^{-1}, \\ \tilde{Q} &= -P^{-1}QM^{-1}, \qquad\qquad \text{where } M = S - Q^T P^{-1} Q \\ \tilde{S} &= M^{-1} \quad . \end{aligned} \tag{A.4}$$

There are also equivalent formulae

$$\begin{aligned} \tilde{P} &= N^{-1}, \\ \tilde{Q} &= -N^{-1}QS^{-1}, \qquad\qquad \text{where } N = P - QS^{-1}Q^T \\ \tilde{S} &= S^{-1} + S^{-1}Q^T N^{-1} Q S^{-1} \quad . \end{aligned} \tag{A.5}$$

A.2 Product of Gaussians

When using linear models with Gaussian priors, the likelihood and the prior are both Gaussian. Their product is proportional to the posterior (also Gaussian), and their integral is equal to the marginal likelihood (or evidence). Consider the random vector $\mathbf{x}$ of size $n \times 1$ and the following product:

$$\mathcal{N}(\mathbf{x}|\mathbf{a}, A)\,\mathcal{N}(P\,\mathbf{x}|\mathbf{b}, B) = z_c\,\mathcal{N}(\mathbf{x}|\mathbf{c}, C) \quad , \tag{A.6}$$

where $\mathcal{N}(\mathbf{x}|\mathbf{a}, A)$ denotes the probability of $\mathbf{x}$ under a Gaussian distribution centered on $\mathbf{a}$ (of size $n \times 1$) and with covariance matrix A (of size $n \times n$). P is a matrix of size $n \times m$ and vectors $\mathbf{b}$ and $\mathbf{c}$ are of size $m \times 1$, and matrices B and C of size $m \times m$. The product of two Gaussians is proportional to a new Gaussian with covariance and mean given by:

$$C = \left(A^{-1} + P\,B^{-1}P^\top\right)^{-1} \quad , \qquad c = C\,\left(A^{-1}\mathbf{a} + P\,B\,\mathbf{b}\right) \quad .$$

The normalizing constant z_c is gaussian in the means $\mathbf{a}$ and $\mathbf{b}$ of the two Gaussians that form the product on the right side of (A.6):

$$\begin{aligned} z_c = {} & (2\,\pi)^{-\frac{m}{2}} |B + P^\top A\,P|^{-\frac{1}{2}} \\ & \times \exp\left(-\frac{1}{2}(\mathbf{b} - P\,\mathbf{a})^\top \left(B + P^\top A\,P\right)^{-1} (\mathbf{b} - P\,\mathbf{a})\right) \quad . \end{aligned}$$

A.3 Incremental Cholesky Factorisation

Consider the quadratic form:

$$Q(\boldsymbol{\alpha}) = -\mathbf{v}^\top \boldsymbol{\alpha} + \frac{1}{2}\,\boldsymbol{\alpha}^\top A\,\boldsymbol{\alpha} \ , \tag{A.7}$$

where A is a symmetric positive definite matrix of size $n \times n$ and $\mathbf{v}$ is a vector of size $n \times 1$. Suppose we have already obtained the minimum and the minimizer of $Q(\boldsymbol{\alpha})$, given by:

$$Q_{opt} = -\frac{1}{2}\,\mathbf{v}^\top A^{-1}\,\mathbf{v} \ , \qquad \boldsymbol{\alpha}_{opt} = A^{-1}\,\mathbf{v} \ . \tag{A.8}$$

We now want to minimize an augmented quadratic form $Q_i(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is now of size $n \times 1$ and A and $\mathbf{v}$ are replaced by A_i and $\mathbf{v}_i$ of size $n+1 \times n+1$ and $n+1 \times 1$ respectively, given by:

$$A_i = \begin{bmatrix} A & \mathbf{b}_i \\ \mathbf{b}_i^\top & c_i \end{bmatrix} \ , \qquad \mathbf{v}_i = \begin{bmatrix} \mathbf{v} \\ v_i \end{bmatrix} \ .$$

Assume that vector $\mathbf{b}_i$ of size $n \times 1$ and scalars c_i and v_i are somehow obtained. We want to exploit the incremental nature of A_i and $\mathbf{v}_i$ to reduce the number of operations necessary to minimize $Q_i(\boldsymbol{\alpha})$. One option would be to compute A_i^{-1} using inversion by partitioning, with cost $\mathcal{O}\left((n+1)^2\right)$ if A^{-1} is known. For iterated incremental computations, using the Cholesky decomposition of A_i is numerically more stable. Knowing L, the Cholesky decomposition of A, the Cholesky decomposition L_i of A_i can be computed as:

$$L_i = \begin{bmatrix} L & 0 \\ \mathbf{z}_i^\top & d_i \end{bmatrix} \ , \qquad L\,\mathbf{z}_i = \mathbf{b}_i, \qquad d_i^2 = c_i - \mathbf{z}_i^\top \mathbf{z}_i \ . \tag{A.9}$$

The computational cost is $\mathcal{O}(n^2/2)$, corresponding to the computation of $\mathbf{z}_i$ by back-substitution. Q_i^{min} can be computed as:

$$Q_i^{min} = Q^{min} - \frac{1}{2}\,u_i^2 \ , \qquad u_i = \frac{1}{d_i}\,(v_i - \mathbf{z}_i^\top \mathbf{u}) \ , \qquad L\,\mathbf{u} = \mathbf{v} \ , \tag{A.10}$$

and the minimizer $\boldsymbol{\alpha}^{opt}$ is given by:

$$L^\top\,\boldsymbol{\alpha}^{opt} = \mathbf{u}_i \ , \qquad \mathbf{u}_i = \begin{bmatrix} \mathbf{u} \\ u_i \end{bmatrix} \ . \tag{A.11}$$

Notice that knowing $\mathbf{u}$ from the previous iteration, computing Q_i^{min} has a cost of $\mathcal{O}(n)$. This is interesting if many different i's need to be explored, for which only the minimum of Q_i is of interest, and not the minimizer. Once the optimal

i has been found, computing the minimizer $\boldsymbol{\alpha}^{opt}$ requires a back-substitution, with a cost of $\mathcal{O}(n^2/2)$.

It is interesting to notice that as a result of computing L_i one obtains "for free" the determinant of A_i (an additional cost of $\mathcal{O}(m)$ to th e$\mathcal{O}(nm)$ cost of the incremental Cholesky). In Sect. A.4 we give a general expression of incremental determinants.

A.4 Incremental Determinant

Consider a square matrix A_i that has a row and a column more than square matrix A of size $n \times n$:

$$A_i = \begin{bmatrix} A & \mathbf{b}_i \\ \mathbf{c}_i^\top & d_i \end{bmatrix} \quad . \tag{A.12}$$

The determinant of A_i is given by

$$|A_i| = |A| \cdot (d_i - \mathbf{b}_i^\top A^{-1} \mathbf{c}_i) \quad . \tag{A.13}$$

In the interesting situation where A^{-1} is known, the new determinant is computed at a cost of $\mathcal{O}(m^2)$.

A.5 Derivation of (3.29)

We give here details of the needed algebra for computing the predictive distribution of the Reduced Rank Gaussian Process. Recall that at training time we use a finite linear model approximation, with less weights than training inputs. Each weight has an associated support input possibly selected from the training inputs. The linear model and prior on the weights are:

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} = \Phi_{nm} \cdot \begin{bmatrix} \boldsymbol{\alpha} \\ \alpha_* \end{bmatrix} \quad , \qquad p\left(\begin{bmatrix} \boldsymbol{\alpha} \\ \alpha_* \end{bmatrix} \middle| \mathbf{x}_*, X, \theta \right) \sim \mathcal{N}\left(0, A^{-1}\right) \quad .$$

where we have defined

$$\Phi_{nm} = \begin{bmatrix} \mathbf{K}_{nm} & \mathbf{k}_* \\ \mathbf{k}(\mathbf{x}_*)^\top & k_{**} \end{bmatrix} \quad , \qquad A = \begin{bmatrix} \mathbf{K}_{mm} & \mathbf{k}(\mathbf{x}_*) \\ \mathbf{k}(\mathbf{x}_*)^\top & k_{**} \end{bmatrix} \quad . \tag{A.14}$$

The induced prior over functions is Gaussian with mean zero and covariance matrix C:

$$p\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \middle| \mathbf{x}_*, X, \theta \right) \sim \mathcal{N}(0, C) \quad , \qquad C = \Phi_{nm} \, A^{-1} \, \Phi_{nm}^\top \quad . \tag{A.15}$$

We use inversion by partitioning to compute A^{-1}:

$$\begin{aligned} A^{-1} &= \left[\begin{array}{cc} \mathbf{K}_{mm}^{-1} + \mathbf{K}_{mm}^{-1}\mathbf{k}(\mathbf{x}_*)\,\mathbf{k}(\mathbf{x}_*)^\top\mathbf{K}_{mm}^{-1} & -\mathbf{K}_{mm}^{-1}\mathbf{k}(\mathbf{x}_*)/c_* \\ -\mathbf{k}(\mathbf{x}_*)^\top\mathbf{K}_{mm}^{-1}/c_* & 1/c_* \end{array} \right] \ , \\ c_* &= k_{**} - \mathbf{k}(\mathbf{x}_*)^\top\mathbf{K}_{mm}^{-1}\mathbf{k}(\mathbf{x}_*) \ , \end{aligned}$$

which allows to obtain C:

$$C = \left[\begin{array}{cc} C_{nn} & \mathbf{k}_* \\ \mathbf{k}_*^\top & k_{**} \end{array} \right] \ , \qquad C_{nn} \equiv \mathbf{K}_{nm}\,\mathbf{K}_{mm}^{-1}\,\mathbf{K}_{nm}^\top + \mathbf{v}_*\mathbf{v}_*^\top/c_* \ , \tag{A.16}$$

where $\mathbf{v}_* \equiv \mathbf{k}_* - \mathbf{K}_{nm}\,\mathbf{K}_{mm}^{-1}\,\mathbf{k}(\mathbf{x}_*)$. We can now compute the distribution of f_* conditioned $\mathbf{f}$:

$$p(f_*|\mathbf{f}, \mathbf{x}_*, X, \theta) \sim \mathcal{N}\left(\mathbf{k}_*^\top C_{nn}^{-1}\mathbf{f}, k_{**} - \mathbf{k}_*^\top C_{nn}^{-1}\mathbf{k}_*\right) \ . \tag{A.17}$$

The predictive distribution, obtained as in (3.5), is Gaussian with mean and variance given by (3.29). We repeat their expressions here for convenience:

$$\begin{aligned} m_*(\mathbf{x}_*) &= \mathbf{k}_*^\top \left[\mathbf{K}_{nm}\,\mathbf{K}_{mm}^{-1}\,\mathbf{K}_{nm}^\top + \sigma^2\,\mathbf{I} + \mathbf{v}_*\mathbf{v}_*^\top/c_*\right]^{-1}\mathbf{y} \ , \\ v_*(\mathbf{x}_*) &= \sigma^2 + k_{**} + \mathbf{k}_*^\top \left[\mathbf{K}_{nm}\,\mathbf{K}_{mm}^{-1}\,\mathbf{K}_{nm}^\top + \sigma^2\,\mathbf{I} + \mathbf{v}_*\mathbf{v}_*^\top/c_*\right]^{-1}\mathbf{k}_* \ . \end{aligned}$$

A.6 Matlab Code for the RRGP

We believe that one very exciting part of looking at a new algorithm is "trying it out"! We would like the interested reader to be able to train our Reduced Rank Gaussian Process (RRGP) algorithm. Training consists in finding the value of the hyperparameters that minimizes the negative log evidence of the RRGP (we give it in Sect. 3.3.1). To do this we first need to be able to compute the negative log evidence and its derivatives with respect to the hyperparameters. Then we can plug this to a gradient descent algorithm to perform the actual learning.

We give a Matlab function, `rrgp_nle`, that computes the negative log evidence of the RRGP and its derivatives for the squared exponential covariance function (given in (3.1)). The hyperparameters of the squared exponential covariance function are all positive. To be able to use unconstrained optimization, we optimize with respect to the logarithm of the hyperparameters.

An auxiliary Matlab function `sq_dist` is needed to compute squared distances. Given to input matrices of sizes $d \times n$ and $d \times m$, the function returns the $n \times m$ matrix of squared distances between all pairs of columns from the inputs matrices. The authors would be happy to provide their own Matlab MEX implementation of this function upon request.

A.6.1 Inputs to the Function rrgp_nle:

- X: $D+2\times 1$ vector of log hyperparameters, $\mathtt{X} = [\log\theta_1, \dots \log\theta_{D+1}, \log\sigma]^\top$, see (3.1)
- input: $n \times D$ matrix of training inputs
- target: $n \times 1$ matrix of training targets
- m: scalar, size of the support set

A.6.2 Outputs of the Function rrgp_nle:

- f: scalar, evaluation of the negative log evidence at X
- f: $D + 2 \times 1$ vector of derivatives of the negative log evidence evaluated at X

A.6.3 Matlab Code of the Function rrgp_nle:

```
function [f,df] = rrgp_nle(X,input,target,m)

% number of examples and dimension of input space
[n, D] = size(input);
input = input ./ repmat(exp(X(1:D))',n,1);

% write the noise-free covariance of size n x m
Knm = exp(2*X(D+1))*exp(-0.5*sq_dist(input',input(1:m,:)'));
% add little jitter to Kmm part
Knm(1:m,:) = Knm(1:m,:)+1e-8*eye(m);

Cnm = Knm/Knm(1:m,:);
Smm = Knm'*Cnm + exp(2*X(D+2))*eye(m);
Pnm = Cnm/Smm;
wm = Pnm'*target;

% compute function evaluation
invQt = (target-Pnm*(Knm'*target))/exp(2*X(D+2));
logdetQ = (n-m)*2*X(D+2) + sum(log(abs(diag(lu(Smm)))));
f = 0.5*logdetQ + 0.5*target'*invQt + 0.5*n*log(2*pi);

% compute derivatives
```

```
df = zeros(D+2,1);

for d=1:D
  Vnm = -sq_dist(input(:,d)',input(1:m,d)').*Knm;
  df(d) = (invQt'*Vnm)*wm - 0.5*wm'*Vnm(1:m,:)*wm+...
  -sum(sum(Vnm.*Pnm))+0.5*sum(sum((Cnm*Vnm(1:m,:)).*Pnm));
end
aux = sum(sum(Pnm.*Knm));
df(D+1) = -(invQt'*Knm)*wm+aux;
df(D+2) = (n-aux) - exp(2*X(D+2))*invQt'*invQt;
```

Appendix B

Time Series Prediction Based on the Relevance Vector Machine with Adaptive Kernels

---

In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2002*, volume 1, pages 985-988, Piscataway, New Jersey, IEEE. This paper was awarded an oral presentation.

The experimental section of the paper has been extended in Sect. 2.3 to non-isotropic basis functions, which allow a significant increase in predictive power.

TIME SERIES PREDICTION BASED ON THE RELEVANCE VECTOR MACHINE WITH ADAPTIVE KERNELS

Joaquin Quiñonero-Candela and Lars Kai Hansen

Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, Building 321, DK-2800 Kongens Lyngby, Denmark
{jqc,lkhansen}@imm.dtu.dk

ABSTRACT

The *Relevance Vector Machine* (RVM) introduced by Tipping is a probabilistic model similar to the widespread *Support Vector Machines* (SVM), but where the training takes place in a Bayesian framework, and where predictive distributions of the outputs instead of point estimates are obtained. In this paper we focus on the use of RVM's for regression. We modify this method for training generalized linear models by adapting automatically the width of the basis functions to the optimal for the data at hand. Our Adaptive RVM is tried for prediction on the chaotic Mackey-Glass time series. Much superior performance than with the standard RVM and than with other methods like neural networks and local linear models is obtained.

1. INTRODUCTION

Generalized linear models perform a nonlinear projection of the input space into a transformed space by means of a set of nonlinear basis functions. A pure linear model is then applied to the transformed space, whose dimension is equal to the number of nonlinear basis functions. Given an input $\mathbf{x}$, the output of the generalized linear model is given by

$$y(\mathbf{x}) = \sum_{j=1}^{M} \omega_j \, \phi_j(\mathbf{x}) + \omega_0 \tag{1}$$

where $\{\phi_j\}$ are the nonlinear basis functions and $\{\omega_j\}$ are the model 'weights'. Unlike in the *Support Vector Machines* (SVM) framework where the basis functions must satisfy Mercer's kernel theorem, in the RVM case there is no restriction on the basis functions [1, 2]. In our case, the basis functions are chosen as Gaussians centered on each of the training points. The model we use can be seen as a particular case of a single hidden layer RBF network with Gaussian radial basis functions centered on the training points.

This work is funded by the EU ***Multi-Agent Control*** Research Training Network - EC TMR grant HPRNCT-1999-00107.

Like SVM's, RVM's yield a sparse solution, i.e., the model is built on a few 'key' training vectors only (like a pruned version of the particular RBF network). But as in the SVM case, no optimization of the basis functions is performed along with the training of the model weights. We propose a modification of the RVM algorithm that includes the optimization of the basis functions, in particular of the variance of the Gaussian functions that we use. We will show that our Adaptive RVM allows the model to be virtually non-parametric, while the performance of basic RVM's depends dramatically on a good choice of the parameters of the basis functions.

In the next section, we summarize the Bayesian framework used to train RVM's, and in Section 3 we highlight the importance of adapting the basis functions and present our improvement to the RVM. Finally, we compare the Adaptive RVM algorithm with other methods for predicting the Mackey-Glass chaotic time series.

2. THE RELEVANCE VECTOR MACHINE

Once the basis functions of the model described in equation (1) are defined, a maximum likelihood approach like the normal equations could be used for training the model weights $\{\omega_j\}$. Training such a flexible linear model, with as many parameters (weights) as training examples using maximum likelihood leads to over-fitting. Generalization capability can be pursued by doing the training in a Bayesian framework.

Rather than attempting to make point predictions of the optimal value of the model weight parameters, a *prior* distribution is defined over each of the weights. In the RVM framework, Gaussian prior distributions are chosen:

$$p(\omega_j|\alpha_j) = \sqrt{\frac{\alpha_j}{2\pi}} \exp\left(-\frac{1}{2}\alpha_j \, \omega_j^2\right) \tag{2}$$

where α_j is the *hyperparameter* that governs the prior defined over the weight ω_j.

Given a set of input-target training pairs $\{x_i, t_i\}_{i=1}^N$, assuming that the targets are independent and that the noise of the data is Gaussian with variance σ^2, the likelihood of the training set can be written as

$$p(\mathbf{t}|\boldsymbol{\omega},\sigma^2) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{t}-\boldsymbol{\Phi}\boldsymbol{\omega}\|^2\right) \quad (3)$$

where $\mathbf{t} = (t_1,\ldots,t_N)^T$, $\boldsymbol{\omega} = (\omega_1,\ldots,\omega_N)^T$ and $\boldsymbol{\Phi}$ is a matrix whose rows contain the response of all basis functions to the inputs $(\boldsymbol{\Phi})_{i,:} = [1,\phi_1(\mathbf{x}_i),\ldots,\phi_N(\mathbf{x}_i)]$.

With the prior and the likelihood distributions, the *posterior* distribution over the weights can be computed using Bayes rule

$$p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha},\sigma^2) = \frac{p(\mathbf{t}|\boldsymbol{\omega},\sigma^2)p(\boldsymbol{\omega}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2)} \quad (4)$$

where $\boldsymbol{\alpha} = (\alpha_0,\ldots,\alpha_N)^T$. The resulting posterior distribution over the weights is the multi-variate Gaussian distribution

$$p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha},\sigma^2) = \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}) \quad (5)$$

where the covariance and the mean are respectively given by:

$$\begin{aligned} \boldsymbol{\Sigma} &= (\sigma^{-2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\mathbf{A})^{-1} && (6)\\ \boldsymbol{\mu} &= \sigma^{-2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{t} && (7) \end{aligned}$$

with $\mathbf{A} = \mathrm{diag}(\alpha_0,\ldots,\alpha_N)$.

The likelihood distribution over the training targets, given by equation (3), can be "marginalized" by integrating out the weights:

$$p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) = \int p(\mathbf{t}|\boldsymbol{\omega},\sigma^2)\,p(\boldsymbol{\omega}|\boldsymbol{\alpha})\,d\boldsymbol{\omega} \quad (8)$$

to obtain the *marginal likelihood* for the hyperparameters:

$$p(\mathbf{t}|\boldsymbol{\alpha},\sigma^2) = \mathcal{N}(0,\mathbf{C}) \quad (9)$$

where the covariance is given by $\mathbf{C} = \sigma^2\mathbf{I}+\boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T$.

In the RVM scheme, the estimated value of the model weights is given by the mean of the posterior distribution (5), which is also the *maximum a posteriori* (MP) estimate of the weights. The MP estimate of the weights depends on the value of the hyperparameters $\boldsymbol{\alpha}$ and of the noise σ^2. The estimate of these two variables $\hat{\boldsymbol{\alpha}}$ and $\hat{\sigma}^2$ is obtained by maximizing the marginal likelihood (9).

The uncertainty about the optimal value of the weights reflected by the posterior distribution (5) is used to express uncertainty about the predictions made by the model. Given a new input $\mathbf{x}_*$, the probability distribution of the corresponding output is given by the *predictive distribution*

$$\begin{aligned} &p(t_*|\mathbf{x}_*,\hat{\boldsymbol{\alpha}},\hat{\sigma}^2) = \\ &\qquad \int p(t_*|\mathbf{x}_*,\boldsymbol{\omega},\hat{\sigma}^2)\,p(\omega|\mathbf{t},\hat{\boldsymbol{\alpha}},\hat{\sigma}^2)\,d\boldsymbol{\omega} \end{aligned} \quad (10)$$

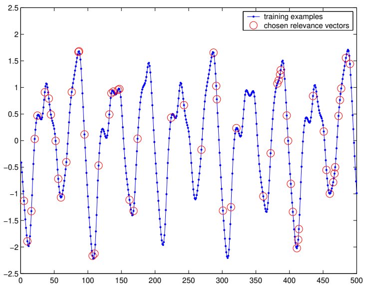

Fig. 1. Relevance Vectors chosen from the training set to build a generalized linear model for prediction.

which has the Gaussian form

$$p(t_*|\mathbf{x}_*,\hat{\boldsymbol{\alpha}},\hat{\sigma}^2) = \mathcal{N}(y_*,\sigma_*^2) \quad (11)$$

where the mean and the variance (*uncertainty*) of the prediction are respectively

$$\begin{aligned} y_* &= (\boldsymbol{\Phi})_{i,:}\,\boldsymbol{\mu} && (12)\\ \sigma_*^2 &= \hat{\sigma}^2 + (\boldsymbol{\Phi})_{i,:}\,\Sigma\,(\boldsymbol{\Phi})_{i,:}^T && (13) \end{aligned}$$

The maximization of the marginal likelihood (9) with respect to $\boldsymbol{\alpha}$ and σ^2 is performed iteratively, as there is no closed solution [1]. In practice, during the iterative re-estimation many of the hyperparameters α_j approach infinity, yielding a posterior distribution (5) of the corresponding weight ω_j that tends to be a delta function centered around zero. The corresponding weight is thus deleted from the model, as well as its associated basis function $\phi_j(\mathbf{x})$. In the RVM framework, each basis function $\phi_j(\mathbf{x})$ is associated to (or centered around) a training example $\mathbf{x}_j$ so that $\phi_j(\mathbf{x}) = g(\mathbf{x}_j,\mathbf{x})$. The model is built on the few training examples whose associated hyperparameters do not go to infinity during the training process, leading to a sparse solution. These remaining examples are called the *Relevance Vectors* (RV).

We here want to examine the RVM approach for time series prediction. We choose a hard prediction problem, the MacKey-Glass chaotic time series, which is well-known for its strong non-linearity. Optimized non-linear models can have a prediction error which is three orders of magnitude lower than an optimized linear model [3]. Figure 1 shows a piece of the chaotic time series and we have furthermore marked the training targets associated to the RV's extracted from a training set composed by 500 samples of the Mackey-Glass chaotic time series.

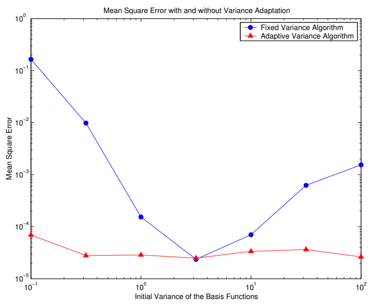

Fig. 2. Prediction mean square error with and without adapting the variance of the basis functions.

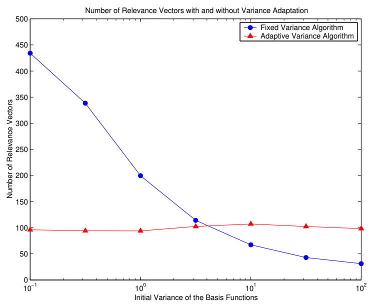

Fig. 3. Number of RV's selected with and without adapting the basis functions with respect to their initial width.

The Mackey-Glass attractor is a non-linear chaotic system described by the following equation:

$$\frac{dz(t)}{dt} = -bz(t) + a\frac{z(t-\tau)}{1+z(t-\tau)^{10}} \qquad (14)$$

where the constants are set to $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is resampled with period 1 according to standard practice. The inputs are formed by $L = 16$ samples spaced 6 periods from each other $\mathbf{x}_k = [z(k-6), z(k-12), \ldots, z(k-6L)]$ and the targets are chosen to be $t_k = z(k)$ to perform six steps ahead prediction [3].

The standard RVM approach is used, with Gaussian basis functions of fixed variance $\nu^2 = 5$.

3. ADAPTING THE BASIS FUNCTIONS

In the training process of a generalized linear model (1) under the RVM scheme described in the previous section, only the weights and hyperparameters are optimized. It is assumed that the basis functions are given. Yet the performance of the model depends dramatically on the choice of the basis functions and the value of their parameters. In the work presented in this paper the basis functions are isotropic Gaussian functions of the same variance, one centered on each training point. The variance is held constant in the conventional RVM approach, while we optimize it in the Adaptive RVM.

The importance of the kernel width parameter is illustrated in Figure 2. We build a generalized linear model (1), that we train using both the conventional RVM scheme, and our adaptive version of it for a time series prediction problem. We here use 700 training examples, and a large set of 8500 test examples to monitor performance. The upper curve in Figure 2 shows the mean square error obtained by training the RVM for a set of increasing widths of the basis functions. Each experiment is repeated 10 times: average values are represented. We note that the performance heavily depends on the width of the basis functions. The similar experiment using the adaptive scheme, described below, where the variance is optimized from variable initial values, systematically improves performance relative to the fixed variance case.

For a given number of training examples, the number of RV chosen depends on the variance of the basis functions. Figure 3 shows the number of RV's chosen as a function of the initial variance both for the conventional and the adaptive approaches. Our adaptive approach selects the number of RV's that allows the best performance, independently of the initial value of the basis functions' variance.

The RVM method iteratively maximizes the marginal likelihood (9) with respect to the hyper-parameters $\boldsymbol{\alpha}$ and to the noise σ^2. We can re-write the marginal likelihood to explicitly condition it on the variance ν^2 of the Gaussian basis functions

$$p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2, \nu^2) = \mathcal{N}(0, \sigma^2\mathbf{I} + \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^{\mathbf{T}}) \qquad (15)$$

which depends on ν^2 through the basis functions matrix $\boldsymbol{\Phi}$.

In our approach, we maximize (15) with respect to ν^2 at each iteration. This is done by maximizing the logarithm of the marginal likelihood. As the width of the basis functions is equal for all, we have to solve a 1D search problem. Evaluating the derivative of the logarithm of (15) with respect to ν^2 is computationally much more expensive than just evaluating the marginal likelihood, hence we decided to use a direct search method due to Hooke and Jeeves [4].

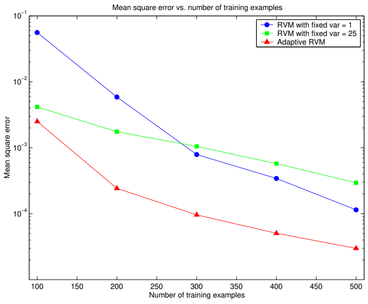

Fig. 4. Prediction mean square error as a function of the number of training examples, for a big and a small value of the variance for the conventional RVM and for the Adaptive RVM.

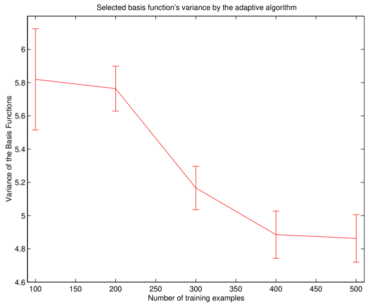

Fig. 5. Value of the variance ν^2 chosen by the Adaptive RVM for different numbers of training examples.

From Figure 2 it appears clearly that for a given number of training examples, there exists an optimal value the basis function width ν^2. But this optimal value depends on the number of training examples, as can be seen from Figure 4. While the conventional RVM performs well for the number of training examples that suits it's fixed ν^2, our approach adapts ν^2 to an optimal value. Figure 5 illustrates how the optimal value of ν^2 decreases for larger training sets, the number of RV's was also found to increase (data not shown).

	Train	**Test**
Simple linear model	9.7×10^{-2}	9.6×10^{-2}
5 nearest-neighbors	4.8×10^{-7}	8.4×10^{-5}
Pruned network	3.1×10^{-5}	3.4×10^{-5}
Adaptive RVM	2.3×10^{-6}	5.5×10^{-6}

Table 1. Training and test mean square prediction error for the Mackey-Glass chaotic time series.

We compare our Adaptive RVM with a simple linear model, with a 5 nearest-neighbors local linear model and with the pruned neural network used in [3] for 6 steps ahead prediction. The training set contains 1000 examples, and the test set 8500 examples. Average values of 10 repetitions are presented. The Adaptive RVM uses an average of 108 RV's in this example. It is remarkable that the Adaptive RVM so clearly outperforms a carefully optimized MLP, we currently investigate other time series prediction problems in order to test the hypothesis that highly non-linear problems are better modeled by non-parametric models with Bayesian complexity control.

4. CONCLUSIONS

Sparse generalized linear models like the RVM (and SVM's) present excellent performance on time series prediction, but are severely limited by the manual choice of the parameters of the basis functions. To overcome this limitation, we propose the Adaptive RVM that automatically optimizes the parameters of the basis functions. The resulting time series predictor outperforms a carefully optimized artificial neural network. The approach can be generalized to locally adapt the kernel widths yielding an even more flexible predictor, however, optimization then becomes non-trivial.

Acknowledgements We would like to thank Carl E. Rasmussen, Michael Saunders and Hans Bruun Nielsen for useful discussion.

5. REFERENCES

[1] Michael E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[2] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[3] Claus Svarer, Lars K. Hansen, Jan Larsen, and Carl E. Rasmussen, "Designer networks for time series processing," *Proceedings of the III IEEE Workshop on Neural Networks for Signal Processing*, pp. 78–87, 1993.

[4] R. Hooke and T. A. Jeeves, " "direct search" solution of numerical and statistical problems," *J. Assoc. Comput.*, pp. 212–229, March 1961.

Appendix C

Incremental Gaussian Processes

In Becker, S., Thrun, S., and Obermayer, L., editors, *Advances in Neural Information Processing Systems 15*, pages 1001–1008, Cambridge, Massachussetts. MIT Press.

In Sect. 2.4 we have extended the algorithmic analysis of the incremental method, and we have proposed to fix the size of the active set.

Incremental Gaussian Processes

Joaquin Quiñonero-Candela
Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark
jqc@imm.dtu.dk

Ole Winther
Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark
owi@imm.dtu.dk

Abstract

In this paper, we consider Tipping's relevance vector machine (RVM) [1] and formalize an incremental training strategy as a variant of the expectation-maximization (EM) algorithm that we call subspace EM. Working with a subset of active basis functions, the sparsity of the RVM solution will ensure that the number of basis functions and thereby the computational complexity is kept low. We also introduce a mean field approach to the intractable classification model that is expected to give a very good approximation to exact Bayesian inference and contains the Laplace approximation as a special case. We test the algorithms on two large data sets with $\mathcal{O}(10^3 - 10^4)$ examples. The results indicate that Bayesian learning of large data sets, e.g. the MNIST database is realistic.

1 Introduction

Tipping's relevance vector machine (RVM) both achieves a sparse solution like the support vector machine (SVM) [2, 3] and the probabilistic predictions of Bayesian kernel machines based upon a Gaussian process (GP) priors over functions [4, 5, 6, 7, 8]. Sparsity is interesting both with respect to fast training and predictions and ease of interpretation of the solution. Probabilistic predictions are desirable because inference is most naturally formulated in terms of probability theory, i.e. we can manipulate probabilities through Bayes theorem, reject uncertain predictions, etc.

It seems that Tipping's relevance vector machine takes the best of both worlds. It is a GP with a covariance matrix spanned by a small number of basis functions making the computational expensive matrix inversion operation go from $\mathcal{O}(N^3)$, where N is the number of training examples to $\mathcal{O}(M^2N)$ (M being the number of basis functions). Simulation studies have shown very sparse solutions $M \ll N$ and good test performance [1]. However, starting the RVM learning with as many basis functions as examples, i.e. one basis function in each training input point, leads to the same complexity as for Gaussian processes (GP) since in the initial step no basis functions are removed. That lead Tipping to suggest in an appendix in Ref. [1] an incremental learning strategy that starts with only a single basis function and adds basis functions along the iterations. The total number of basis functions is kept low because basis functions are also removed. In this paper we formalize this strategy using straightforward expectation-maximization (EM) [9] arguments to prove that the scheme is the guaranteed convergence to a local maximum of the likelihood of the model parameters.

Reducing the computational burden of Bayesian kernel learning is a subject of current interest. This can be achieved by numerical approximations to matrix inversion [10] and suboptimal projections onto finite subspaces of basis functions without having an explicit parametric form of such basis functions [11, 12]. Using mixtures of GPs [13, 14] to make the kernel function input dependent is also a promising technique. None of the Bayesian methods can currently compete in terms of speed with the efficient SVM optimization schemes that have been developed, see e.g. [3].

The rest of the paper is organized as follows: In section 2 we present the extended linear models in a Bayesian perspective, the regression model and the standard EM approach. In section 3, a variation of the EM algorithm, that we call the *Subspace EM* (SSEM) is introduced that works well with sparse solution models. In section 4, we present the second main contribution of the paper: a mean field approach to RVM classification. Section 5 gives results for the Mackey-Glass time-series and preliminary results on the MNIST hand-written digits database. We conclude in section 6.

2 Regression

An extended linear model is build by transforming the input space by an arbitrary set of *basis functions* $\phi_j : R^D \to R$ that performs a non-linear transformation of the D-dimensional input space. A linear model is applied to the transformed space whose dimension is equal to the number of basis functions M:

$$y(\mathbf{x}_i) = \sum_{j=1}^{M} \omega_j \, \phi_j(\mathbf{x}_i) = \boldsymbol{\Phi}(\mathbf{x}_i) \cdot \boldsymbol{\omega} \tag{1}$$

where $\boldsymbol{\Phi}(\mathbf{x}_i) \equiv [\phi_1(\mathbf{x}_i), \ldots, \phi_M(\mathbf{x}_i)]$ denotes the ith row of the *design matrix* $\boldsymbol{\Phi}$ and $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_N)^T$ is the *weights* vector. The output of the model is thus a linear superposition of completely general basis functions. While it is possible to optimize the parameters of the basis functions for the problem at hand [1, 15], we will in this paper assume that they are given.

The simplest possible regression learning scenario can be described as follows: a set of N input-target training pairs $\{x_i, t_i\}_{i=1}^N$ are assumed to be independent and contaminated with Gaussian noise of variance σ^2. The likelihood of the parameters $\boldsymbol{\omega}$ is given by

$$p(\mathbf{t}|\boldsymbol{\omega}, \sigma^2) = \left(2\pi\sigma^2\right)^{-N/2} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{t} - \boldsymbol{\Phi}\boldsymbol{\omega}\|^2\right) \tag{2}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^T$ is the target vector. Regularization is introduced in Bayesian learning by means of a prior distribution over the weights. In general, the implied prior over functions is a very complicated distribution. However, choosing a Gaussian prior on the weights the prior over functions also becomes Gaussian, i.e. a Gaussian process. For the specific choice of a factorized distribution with variance α_j^{-1}:

$$p(\omega_j|\alpha_j) = \sqrt{\frac{\alpha_j}{2\pi}} \exp\left(-\frac{1}{2}\alpha_j \, \omega_j^2\right) \tag{3}$$

the prior over functions $p(\mathbf{y}|\boldsymbol{\alpha})$ is $\mathcal{N}(0, \boldsymbol{\Phi}\mathbf{A}^{-1}\boldsymbol{\Phi}^T)$, i.e. a Gaussian process with covariance function given by

$$\mathrm{Cov}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{M} \frac{1}{\alpha_k} \phi_k(x_i) \phi_k(x_j) \tag{4}$$

where $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_N)^T$ and $\mathbf{A} = \mathrm{diag}(\alpha_0, \ldots, \alpha_N)$. We can now see how sparseness in terms of the basis vectors may arise: if $\alpha_k^{-1} = 0$ the kth basis vector

$\mathbf{\Phi}_k \equiv [\phi_k(\mathbf{x}_1), \ldots, \phi_k(\mathbf{x}_N)]^T$, i.e. the kth column in the design matrix, will not contribute to the model. Associating a basis function with each input point may thus lead to a model with a sparse representations in the inputs, i.e. the solution is only spanned by a subset of all input points. This is exactly the idea behind the relevance vector machine, introduced by Tipping [16]. We will see in the following how this also leads to a lower computational complexity than using a regular Gaussian process kernel.

The posterior distribution over the weights–obtained through Bayes rule–is a Gaussian distribution

$$p(\boldsymbol{\omega}|\mathbf{t}, \boldsymbol{\alpha}, \sigma^2) = \frac{p(\mathbf{t}|\boldsymbol{\omega}, \sigma^2)p(\boldsymbol{\omega}|\boldsymbol{\alpha})}{p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2)} = \mathcal{N}(\mathbf{t}|\boldsymbol{\mu}, \mathbf{\Sigma}) \tag{5}$$

where $\mathcal{N}(\mathbf{t}|\boldsymbol{\mu}, \mathbf{\Sigma})$ is a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$ evaluated at $\mathbf{t}$. The mean and covariance are given by

$$\boldsymbol{\mu} = \sigma^{-2}\mathbf{\Sigma}\mathbf{\Phi}^T\mathbf{t} \tag{6}$$

$$\mathbf{\Sigma} = (\sigma^{-2}\mathbf{\Phi}^T\mathbf{\Phi} + \mathbf{A})^{-1} \tag{7}$$

The uncertainty about the optimal value of the weights captured by the posterior distribution (5) can be used to build probabilistic predictions. Given a new input $\mathbf{x}_*$, the model gives a Gaussian *predictive distribution* of the corresponding target t_*

$$p(t_*|\mathbf{x}_*, \hat{\boldsymbol{\alpha}}, \hat{\sigma}^2) = \int p(t_*|\mathbf{x}_*, \boldsymbol{\omega}, \hat{\sigma}^2)\, p(\boldsymbol{\omega}|\mathbf{t}, \hat{\boldsymbol{\alpha}}, \hat{\sigma}^2)\, d\boldsymbol{\omega} = \mathcal{N}(t_*|y_*, \sigma_*^2) \tag{8}$$

where

$$y_* = \mathbf{\Phi}(\mathbf{x}_*) \cdot \boldsymbol{\mu} \tag{9}$$

$$\sigma_*^2 = \hat{\sigma}^2 + \mathbf{\Phi}(\mathbf{x}_*) \cdot \mathbf{\Sigma} \cdot \mathbf{\Phi}(\mathbf{x}_*)^T \tag{10}$$

For regression it is natural to use y_* and σ_* as the prediction and the error bar on the prediction respectively. The computational complexity of making predictions is thus $\mathcal{O}(M^2P + M^3 + M^2N)$, where M is the number of selected basis functions (RVs) and P is the number of predictions. The two last terms come from the computation of $\mathbf{\Sigma}$ in eq. (7).

The likelihood distribution over the training targets (2) can be "marginalized" with respect to the weights to obtain the *marginal likelihood*, which is also a Gaussian distribution

$$p(\mathbf{t}|\boldsymbol{\alpha}, \sigma^2) = \int p(\mathbf{t}|\boldsymbol{\omega}, \sigma^2)\, p(\boldsymbol{\omega}|\boldsymbol{\alpha})\, d\boldsymbol{\omega} = \mathcal{N}(\mathbf{t}|0, \sigma^2\mathbf{I} + \mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T)\,. \tag{11}$$

Estimating the hyperparameters $\{\alpha_j\}$ and the noise σ^2 can be achieved by maximizing (11). This is naturally carried out in the framework of the expectation-maximization (EM) algorithm since the sufficient statistics of the weights (that act as hidden variables) are available for this type of model. In other cases e.g. for adapting the length scale of the kernel [4], gradient methods have to be used. For regression, the E-step is exact (the lower bound on the marginal likelihood is made equal to the marginal likelihood) and consists in estimating the mean and variance (6) and (7) of the posterior distribution of the weights (5). For classification, the E-step will be approximate. In this paper we present a mean field approach for obtaining the sufficient statistics.

The M-step corresponds to maximizing the expectation of the log marginal likelihood with respect to the posterior, with respect to σ^2 and $\boldsymbol{\alpha}$, which gives the following update rules: $\alpha_j^{new} = \frac{1}{\langle\omega_j^2\rangle}_{p(\boldsymbol{\omega}|\mathbf{t},\boldsymbol{\alpha},\sigma^2)} = \frac{1}{\mu_i^2+\Sigma_{jj}}$, and $(\sigma^2)^{new} = \frac{1}{N}(||\mathbf{t} - \mathbf{\Phi}\,\mu||^2 + (\sigma^2)^{old}\sum_j \gamma_j)$, where the quantity $\gamma_j \equiv 1 - \alpha_j\Sigma_{jj}$ is a measure of how "well-determined" each weight ω_j is by the data [17, 1]. One can obtain a different update rule that gives faster convergence.

```
1.  Set α_j = L for all j. (L is a very large number) Set n = 1
2.  Update the set of active indexes R_n
3.  Perform an E-step in subspace ω_j such that j ∈ R_n
4.  Perform the M-step for all α_j such that j ∈ R_n
5.  If visited all basis functions, end, else go to 2.
```

Figure 1: Schematics of the SSEM algorithm.

Although it is suboptimal in the EM sense, we have never observed it decrease the lower bound on the marginal log-likelihood. The rule, derived in [1], is obtained by differentiation of (11) and by an arbitrary choice of independent terms as is done by [17]. It makes use of the terms $\{\gamma_j\}$:

$$\alpha_j^{new} = \frac{\gamma_j}{\mu_j^2} \qquad (\sigma^2)^{new} = \frac{||\mathbf{t} - \mathbf{\Phi}\,\mu||^2}{N - \sum_j \gamma_j}\,. \tag{12}$$

In the optimization process many α_j grow to infinity, which effectively deletes the corresponding weight and basis function. Note that the EM update and the Mackay update for α_j only implicitly depend upon the likelihood. This means that it is also valid for the classification model we shall consider below.

A serious limitation of the EM algorithm and variants for problems of this type is that the complexity of computing the covariance of the weights (7) in the E-step is $O(M^3 + M^2N)$. At least in the first iteration where no basis functions have been deleted $M = N$ and we are facing the same kind of complexity explosion that limits the applicability of Gaussian processes to large training set. This has lead Tipping [1] to consider a constructive or incremental training paradigm where one basis function is added before each E-step and since basis functions are removed in the M-step, it turns out in practice that the total number of basis functions and the complexity remain low. In the following section we introduce a new algorithm that formalizes this procedure that can be proven to increase the marginal likelihood in each step.

3 Subspace EM

We introduce an incremental approach to the EM algorithm, the *Subspace EM* (SSEM), that can be directly applied to training models like the RVM that rely on a linear superposition of completely general basis functions, both for classification and for regression. Instead of starting with a full model, i.e. where all the basis functions are present with finite α values, we start with a fully pruned model with all α_j set to infinity. Effectively, we start with no model. The model is grown by iteratively including some α_j previously set to infinity to the *active set* of α's. The active set at iteration n, R_n, contains the indices of the basis vectors with α less than infinity:

$$\begin{aligned} &R_1 = 1 \\ &R_n = \{i \mid i \in R_{n-1} \wedge \alpha_i \leq \mathrm{L}\} \cup \{n\} \end{aligned} \tag{13}$$

where L is a finite very large number arbitrarily defined. Observe that R_n contains at most one more element (index) than R_{n-1}. If some of the α's indexed by R_{n-1} happen to reach L at the n-th step, R_n can contain less elements than R_{n-1}. In figure 1 we give a schematic description of the SSEM algorithm.

At iteration n the E-step is taken only in the subspace spanned by the weights whose indexes are in R_n. This helps reducing the computational complexity of the M-step to $O(M^3)$, where M is the number of relevance vectors.

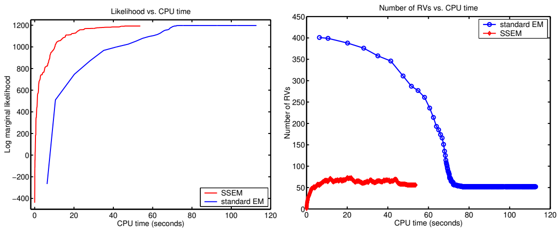

Figure 2: Training on 400 samples of the Mackey-Glass time series, testing on 2000 cases. Log marginal likelihood as a function of the elapsed CPU time (left) and corresponding number of relevance vectors (right) for both SSEM and EM.

Since the initial value of α_j is infinity for all j, for regression the E-step yields always an equality between the log marginal likelihood and its lower bound. At any step n, the posterior can be exactly projected on to the space spanned by the weights ω_j such that $j \in R_n$, because the $\alpha_k = \infty$ for all k not in R_n. Hence in the regression case, the SSEM never decreases the log marginal likelihood. Figure 2 illustrates the convergence process of the SSEM algorithm compared to that of the EM algorithm for regression.

Once all the examples have been visited, we switch to the batch EM algorithm *on the active set* until some convergence criteria has been satisfied, for example until the relative increase in the likelihood is smaller than a certain threshold.

4 Classification

Unlike the model discussed above, analytical inference is not possible for classification models. Here, we will discuss a mean field approach initially proposed for Gaussian processes [8] that are readily translated to RVMs. The mean field approach has the appealing features that it retains the computational efficiency of RVMs, is exact for the regression and reduces to the Laplace approximation in the limit where all the variability comes from the prior distribution.

We consider binary $t = \pm 1$ classification using the *probit* likelihood with 'input' noise σ^2

$$p(t|y(\mathbf{x})) = \mathrm{erf}\left(t\frac{y(\mathbf{x})}{\sigma}\right) , \tag{14}$$

where $Dz \equiv e^{-z^2/2}dz/\sqrt{2\pi}$ and $\mathrm{erf}(x) \equiv \int_{-\infty}^{x} Dz$ is an error function (or cumulative Gaussian distribution). The advantage of using this sigmoid rather than the commonly used 0/1-logistic is that we under the mean field approximation can derive an analytical expression for the predictive distribution $p(t_*|\mathbf{x}_*, \mathbf{t}) = \int p(t_*|y)p(y|\mathbf{x}_*, \mathbf{t})dy$ needed for making Bayesian predictions. The central assumption in mean field theory can be boiled down to saying that $p(y|\mathbf{x}_*, \mathbf{t})$ is approximated by a Gaussian distribution. This leads to the following approximation for the predictive distribution

$$p(t_*|\mathbf{x}_*, \mathbf{t}) = \mathrm{erf}\left(t_*\frac{y_*}{\sigma_*}\right) \tag{15}$$

where the mean and variance of $p(y|\mathbf{x}_*, \mathbf{t})$: y_* and σ_*^2 are given by the eqs. (9) and (10). However, the mean and covariance of the weights are no longer found by analytical expressions, but has to be obtained from a set of non-linear mean field equations that also follow from equivalent assumptions of Gaussianity for the training set outputs $y(\mathbf{x}_i)$ in averages over reduced (or cavity) posterior averages.

In the following, we will only state the results which follows from combining the RVM Gaussian process kernel (4) with the results of [8]. The sufficient statistics of the weights are written in terms of a set of $\mathcal{O}(N)$ mean field parameters

$$\boldsymbol{\mu} = \mathbf{A}^{-1}\mathbf{\Phi}^T\boldsymbol{\tau} \tag{16}$$

$$\mathbf{\Sigma} = \left(\mathbf{A} + \mathbf{\Phi}^T\mathbf{\Omega}\mathbf{\Phi}\right)^{-1} \tag{17}$$

where $\tau_i \equiv \frac{\partial}{\partial y_i^c} \ln Z(y_i^c, V_i^c + \sigma^2)$ and

$$Z(y_i^c, V_i^c + \sigma^2) \equiv \int p(t_i|y_i^c + z\sqrt{V_i^c + \sigma^2})\, Dz = \mathrm{erf}\left(t_i \frac{y_i^c}{\sqrt{V_i^c + \sigma^2}}\right) . \tag{18}$$

The last equality holds for the likelihood eq. (14) and y_i^c and V_i^c are the mean and variance of the so called cavity field. The mean value is $y_i^c = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mu - V_i^c\tau_i$. The distinction between the different approximation schemes is solely in the variance V_i^c: $V_i^c = 0$ is the Laplace approximation, $V_i^c = \left[\mathbf{\Phi}\mathbf{A}^{-1}\mathbf{\Phi}^T\right]_{ii}$ is the so called naive mean field theory and an improved estimate is available from the adaptive TAP mean field theory [8]. Lastly, the diagonal matrix $\mathbf{\Omega}$ is the equivalent of the noise variance in the regression model (compare eqs. (17) and (7) and is given by $\Omega_i = -\frac{\partial \tau_i}{\partial y_i^c}/(1+V_i^c\frac{\partial \tau_i}{\partial y_i^c})$. This set of non-linear equations are readily solved (i.e. fast and stable) by making Newton-Raphson updates in $\boldsymbol{\mu}$ treating the remaining quantities as help variables:

$$\Delta\boldsymbol{\mu} = (\mathbf{I} + \mathbf{A}^{-1}\mathbf{\Phi}^T\mathbf{\Omega}\mathbf{\Phi})^{-1}(\mathbf{A}^{-1}\mathbf{\Phi}^T\boldsymbol{\tau} - \boldsymbol{\mu}) = \mathbf{\Sigma}(\mathbf{\Phi}^T\boldsymbol{\tau} - \mathbf{A}\boldsymbol{\mu}) \tag{19}$$

The computational complexity of the E-step for classification is augmented with respect to the regression case by the fact that it is necessary to construct and invert a $M \times M$ matrix usually many times (typically 20), once for each step of the iterative Newton method.

5 Simulations

We illustrate the performance of the SSEM for regression on the Mackey-Glass chaotic time series, which is well-known for its strong non-linearity. Optimized non-linear models can have a prediction error which is three orders of magnitude lower than an optimized linear model [18]. In [15] we showed that the RVM has an order of magnitude superior performance than carefully tuned neural networks for time series prediction on the Mackey-Glass series. The inputs are formed by $L = 16$ samples spaced 6 periods from each other $\mathbf{x}_k = [z(k-6), z(k-12), \ldots, z(k-6L)]$ and the targets are chosen to be $t_k = z(k)$ to perform six steps ahead prediction (see [18] for details). We use Gaussian basis functions of fixed variance $\nu^2 = 10$. The test set comprises 5804 examples.

We perform prediction experiments for different sizes of the training set. We perform in each case 10 repetitions with different partitions of the data sets into training and test. We compare the test error, the number of RVs selected and the computer time needed for the batch and the SSEM method. We present the results obtained with the growth method relative to the results obtained with the batch method in figure 3. As expected, the relative computer time of the growth method compared with the batch method decreases with size of the training set. For a few thousand examples the SSEM method is an order of magnitude faster than the batch method. The batch method proved only to be faster for 100 training

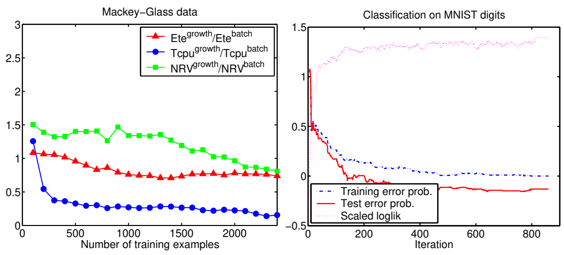

Figure 3: Left: Regression, mean values over 10 repetitions of relative test error, number of RVs and computer time for the Mackey-Glass data, up to 2400 training examples and 5804 test examples. Right: Classification, Log marginal likelihood, test and training errors while training on one class against all the others, 60000 training and 10000 test examples.

examples, and could not be used with data sets of thousands of examples on the machine on which we run the experiments because of its high memory requirements. This is the reason why we only ran the comparison for up to 2400 training example for the Mackey-Glass data set.

To illustrate the performance in classification problems we choose a very large data set, the MNIST database of handwritten digits [19], with 60000 training and 10000 test images. The images are of size 28×28 pixels. We use PCA to project them down to 16 dimensional vectors. We only perform a preliminary experiment consisting of a one against all binary classification problem to illustrate that Bayesian approaches to classification can be used on very large data sets with the SSEM algorithm. We train on 13484 examples (the 6742 *one*'s and another 6742 random non-*one* digits selected at random from the rest) and we use 800 basis functions for both the batch and Subspace EM. In figure 3 we show the convergence of the SSEM in terms of the log marginal likelihood and the training and test probabilities of error. The test probability of error we obtain is 0.74 percent with the SSEM algorithm and 0.66 percent with the batch EM. Under the same conditions the SSEM needed 55 minutes to do the job, while the batch EM needed 186 minutes. The SSEM gives a machine with 28 basis functions and the batch EM one with 31 basis functions. We intend to implement an RVM for multi-class classification and train it on the whole MNIST dataset with the SSEM algorithm, which we estimate will take 3 days in total on a Linux cluster. It is impossible to do such a thing with the batch EM, except on machines capable of inverting 60000×60000 matrices.

6 Conclusion

We have presented a new approach to Bayesian training of linear models, based on a subspace extension of the EM algorithm that we call *Subspace EM* (SSEM). The new method iteratively builds models from a potentially big library of basis functions. It is especially well-suited for models that are constructed such that they yield a sparse solution, i.e. the solution is spanned by small number M of basis functions, which is much smaller than N, the number of examples. A prime example of this is Tipping's relevance vector machine that typically produces solutions that are sparser than those of support vector machines. With the SSEM algorithm the computational complexity and memory requirement decrease from $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ to $\mathcal{O}(M^2N)$ (somewhat higher for classification) and $\mathcal{O}(NM)$. For classification, we have presented a mean field approach that are expected to be a very good approximation to the exact inference and contains the widely used Laplace approximation as an extreme case. We have applied the SSEM algorithm to both a large regression and a large classification data sets. Although preliminary for the latter, we believe that the results demonstrate that Bayesian learning is possible for very large data sets. Similar methods should also be applicable beyond supervised learning.

References

[1] Michael E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[2] Vladimir N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[3] Bernhard Schölkopf and Alex J. Smola, *Learning with Kernels*, MIT Press, Cambridge, 2002.

[4] Carl E. Rasmussen, *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*, Ph.D. thesis, Dept. of Computer Science, University of Toronto, 1996.

[5] Chris K. I. Williams and Carl E. Rasmussen, "Gaussian proceses for regression," in *Advances in Neural Information Processing Systems*, 1996, number 8, pp. 514–520.

[6] David J. C. MacKay, "Gaussian processes - a replacement for supervised neural networks?," Lecture notes for a tutorial in Advances in Neural Information Processing Systems, 1997.

[7] Radford M. Neal, *Bayesian Learning for Neural Networks*, Springer, New York, 1996.

[8] Manfred Opper and Ole Winther, "Gaussian processes for classification: Mean field algorithms," *Neural Computation*, vol. 12, pp. 2655–2684, 2000.

[9] N. M. Dempster, A.P. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. B*, vol. 39, pp. 185–197, 1977.

[10] Chris K. I. Williams and Mathias Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems*, 2001, number 13, pp. 682–688.

[11] Alex J. Smola and Peter L. Bartlett, "Sparse greedy gaussian process regression," in *Advances in Neural Information Processing Systems*, 2001, number 13, pp. 619–625.

[12] Lehel Csató and Manfred Opper, "Sparse representation for gaussian process models," in *Advances in Neural Information Processing Systems*, 2001, number 13, pp. 444–450.

[13] Volker Tresp, "Mixtures of gaussian processes," in *Advances in Neural Information Processing Systems*, 2000, number 12, pp. 654–660.

[14] Carl E. Rasmussen and Zoubin Ghahramani, "Infinite mixtures of gaussian process experts," in *Advances in Neural Information Processing Systems*, 2002, number 14.

[15] Joaquin Quiñonero-Candela and Lars Kai Hansen, "Time series prediction based on the relevance vector machine with adaptive kernels," in *To appear in International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002.

[16] Michael E. Tipping, "The relevance vector machine," in *Advances in Neural Information Processing Systems*, 2000, number 12, pp. 652–658.

[17] David J. C. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

[18] Claus Svarer, Lars K. Hansen, Jan Larsen, and Carl E. Rasmussen, "Designer networks for time series processing," *Proceedings of the III IEEE Workshop on Neural Networks for Signal Processing*, pp. 78–87, 1993.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

Appendix D

Gaussian Process with Uncertain Inputs - Application to Multiple-Step Ahead Time-Series Forecasting

In Becker, S., Thrun, S., and Obermayer, L., editors, *Advances in Neural Information Processing Systems 15*, pages 529–536, Cambridge, Massachussetts. MIT Press. This paper was awarded an oral presentation.

Our contribution to this work was to derive the expressions for exact moment matching. The detailed algebra is given in Sect. 4.1.1, and in an DTU technical report, number IMM-2003-18.

Gaussian Process Priors With Uncertain Inputs – Application to Multiple-Step Ahead Time Series Forecasting

Agathe Girard
Department of Computing Science
University of Glasgow
Glasgow, G12 8QQ
agathe@dcs.gla.ac.uk

Carl Edward Rasmussen
Gatsby Unit
University College London
London, WC1N 3AR
edward@gatsby.ucl.ac.uk

Joaquin Quiñonero Candela
Informatics and Mathematical Modelling
Technical University of Denmark
Richard Petersens Plads, Building 321
DK-2800 Kongens, Lyngby, Denmark
jqc@imm.dtu.dk

Roderick Murray-Smith
Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ
& Hamilton Institute
National University of Ireland, Maynooth
rod@dcs.gla.ac.uk

Abstract

We consider the problem of multi-step ahead prediction in time series analysis using the non-parametric Gaussian process model. k-step ahead forecasting of a discrete-time non-linear dynamic system can be performed by doing repeated one-step ahead predictions. For a state-space model of the form $y_t = f(y_{t-1}, \ldots, y_{t-L})$, the prediction of y at time $t + k$ is based on the point estimates of the previous outputs. In this paper, we show how, using an analytical Gaussian approximation, we can formally incorporate the uncertainty about intermediate regressor values, thus updating the uncertainty on the current prediction.

1 Introduction

One of the main objectives in time series analysis is forecasting and in many real life problems, one has to predict ahead in time, up to a certain time horizon (sometimes called *lead* time or prediction horizon). Furthermore, knowledge of the uncertainty of the prediction is important. Currently, the multiple-step ahead prediction task is achieved by either explicitly training a *direct* model to predict k steps ahead, or by doing repeated one-step ahead predictions up to the desired horizon, which we call the *iterative method*.

There are a number of reasons why the iterative method might be preferred to the 'direct' one. Firstly, the direct method makes predictions for a fixed horizon only, making it computationally demanding if one is interested in different horizons. Furthermore, the larger k, the more training data we need in order to achieve a good predictive performance, because of the larger number of 'missing' data between t and $t+k$. On the other hand, the iterated method provides any k-step ahead forecast, up to the desired horizon, as well as the joint probability distribution of the predicted points.

In the Gaussian process modelling approach, one computes predictive distributions whose means serve as output estimates. Gaussian processes (GPs) for regression have historically been first introduced by O'Hagan [1] but started being a popular non-parametric modelling approach after the publication of [7]. In [10], it is shown that GPs can achieve a predictive performance comparable to (if not better than) other modelling approaches like neural networks or local learning methods. We will show that for a k-step ahead prediction which ignores the accumulating prediction variance, the model is not conservative enough, with unrealistically small uncertainty attached to the forecast. An alternative solution is presented for iterative k-step ahead prediction, with propagation of the prediction uncertainty.

2 Gaussian Process modelling

We briefly recall some fundamentals of Gaussian processes. For a comprehensive introduction, please refer to [5], [11], or the more recent review [12].

2.1 The GP prior model

Formally, the random function, or stochastic process, $f(\mathbf{x})$ is a Gaussian process, with mean $m(\mathbf{x})$ and covariance function $C(\mathbf{x}^p, \mathbf{x}^q)$, if its values at a finite number of points, $f(\mathbf{x}^1), \ldots, f(\mathbf{x}^n)$, are seen as the components of a normally distributed random vector. If we further assume that the process is stationary: it has a constant mean and a covariance function only depending on the distance between the inputs $\mathbf{x}$. For any n, we have

$$f(\mathbf{x}^1), \ldots, f(\mathbf{x}^n) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \,, \tag{1}$$

with $\Sigma_{pq} = \mathrm{Cov}(f(\mathbf{x}^p), f(\mathbf{x}^q)) = C(\mathbf{x}^p, \mathbf{x}^q)$ giving the covariance between the points $f(\mathbf{x}^p)$ and $f(\mathbf{x}^q)$, which is a function of the inputs corresponding to the same cases p and q. A common choice of covariance function is the Gaussian kernel[1]

$$C(\mathbf{x}^p, \mathbf{x}^q) = \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\frac{(\mathbf{x}_d^p - \mathbf{x}_d^q)^2}{w_d^2}\right] \,, \tag{2}$$

where D is the input dimension. The w parameters (correlation length) allow a different distance measure for each input dimension d. For a given problem, these parameters will be adjusted to the data at hand and, for irrelevant inputs, the corresponding w_d will tend to zero.

The role of the covariance function in the GP framework is similar to that of the kernels used in the Support Vector Machines community. This particular choice corresponds to a prior assumption that the underlying function f is *smooth* and *continuous*. It accounts for a high correlation between the outputs of cases with nearby inputs.

[1] This choice was motivated by the fact that, in [8], we were aiming at unified expressions for the GPs and the Relevance Vector Machines models which employ such a kernel. More discussion about possible covariance functions can be found in [5].

2.2 Predicting with Gaussian Processes

Given this prior on the function f and a set of data $\mathcal{D} = \{y^i, \mathbf{x}^i\}_{i=1}^N$, our aim, in this Bayesian setting, is to get the predictive distribution of the function value $f(\mathbf{x}^*)$ corresponding to a new (given) input $\mathbf{x}^*$.

If we assume an additive uncorrelated Gaussian white noise, with variance v_0, relates the targets (observations) to the function outputs, the distribution over the targets is Gaussian, with zero mean and covariance matrix such that $K_{pq} = \Sigma_{pq} + v_0\delta_{pq}$. We then adjust the vector of hyperparameters $\Theta = [w_1 \ldots w_D \; v_1 \; v_0]^T$ so as to maximise the log-likelihood $\mathcal{L}(\Theta) = \log p(\mathbf{y}|\Theta)$, where $\mathbf{t}$ is the vector of observations.

In this framework, for a new $\mathbf{x}^*$, the predictive distribution is simply obtained by conditioning on the training data. The joint distribution of the variables being Gaussian, this conditional distribution, $p(f(\mathbf{x})^*|\mathbf{x}^*, \mathcal{D})$is also Gaussian with mean and variance

$$\mu(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^T \, \mathbf{K}^{-1} \, \mathbf{y} \tag{3}$$

$$\sigma^2(\mathbf{x}^*) = k(\mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T \, K^{-1} \, \mathbf{k}(\mathbf{x}^*) \; , \tag{4}$$

where $\mathbf{k}(\mathbf{x}^*) = [C(\mathbf{x}^*, \mathbf{x}^1), \ldots, C(\mathbf{x}^*, \mathbf{x}^N)]^T$ is the $N \times 1$ vector of covariances between the new point and the training targets and $k(\mathbf{x}^*) = C(\mathbf{x}^*, \mathbf{x}^*) = 1$, with $C(.,.)$ as given by (2).

The predictive mean serves as a point estimate of the function output, $\hat{f}(\mathbf{x}^*)$ with uncertainty $\sigma(\mathbf{x}^*)$. And it is also a point estimate for the target, $\hat{y}^*$, with variance $\sigma^2(\mathbf{x}^*) + v_0$.

3 Prediction at a random input

If we now assume that the input distribution is Gaussian, $\mathbf{x}^* \sim \mathcal{N}(\mu_{x^*}, \mathbf{\Sigma}_{x^*})$, the predictive distribution is now obtain by integrating over $\mathbf{x}^*$

$$p(f(\mathbf{x}^*)|\mu_{x^*}, \Sigma_{x^*}) = \int p(f(\mathbf{x}^*)|\mathbf{x}^*, \mathcal{D}) p(\mathbf{x}^*) d\mathbf{x}^* \; , \tag{5}$$

where $p(f(\mathbf{x}^*)|\mathbf{x}^*, \mathcal{D})$ is Normal, as specified by (3) and (4).

3.1 Gaussian approximation

Given that this integral is analytically intractable ($p(f(\mathbf{x}^*)|\mathbf{x}^*)$ is a complicated function of $\mathbf{x}^*$), we opt for an analytical Gaussian approximation and only compute the mean and variance of $p(f(\mathbf{x}^*)|\mu_{x^*}, \Sigma_{x^*})$. Using the law of iterated expectations and conditional variance, the 'new' mean and variance are given by

$$m(\mu_{x^*}, \Sigma_{x^*}) = E_{x^*}[E_{f(x^*)}[f(\mathbf{x}^*)|\mathbf{x}^*]] = E_{x^*}[\mu(\mathbf{x}^*)] \tag{6}$$

$$\begin{aligned} v(\mu_{x^*}, \Sigma_{x^*}) &= E_{x^*}[\mathrm{var}_{f(x^*)}(f(\mathbf{x}^*)|\mathbf{x}^*)] + \mathrm{var}_{x^*}(E_{f(x^*)}[f(\mathbf{x}^*)|\mathbf{x}^*]) \\ &= E_{x^*}[\sigma^2(\mathbf{x}^*)] + \mathrm{var}_{x^*}(\mu(\mathbf{x}^*)) \end{aligned} \tag{7}$$

where E_{x^*} indicates the expectation under x^*.

In our initial development, we made additional approximations ([2]). A first and second order Taylor expansions of $\mu(\mathbf{x}^*)$ and $\sigma^2(\mathbf{x}^*)$ respectively, around μ_{x^*}, led to

$$m(\mu_{x^*}, \Sigma_{x^*}) = \mu(\mu_{x^*}) \tag{8}$$

$$v(\mu_{x^*}, \Sigma_{x^*}) = \sigma^2(\mu_{x^*}) + \frac{1}{2}\mathrm{Tr}\left\{ \left.\frac{\partial^2 \sigma^2(\mathbf{x}^*)}{\partial \mathbf{x}^* \partial \mathbf{x}^{*T}}\right|_{\mathbf{x}^*=\mu_{x^*}} \mathbf{\Sigma}_{x^*} \right\} + \left.\frac{\partial \mu(\mathbf{x}^*)}{\partial \mathbf{x}^*}\right|_{\mathbf{x}^*=\mu_{x^*}}^T \mathbf{\Sigma}_{x^*} \left.\frac{\partial \mu(\mathbf{x}^*)}{\partial \mathbf{x}^*}\right|_{\mathbf{x}^*=\mu_{x^*}} \tag{9}$$

The detailed calculations can be found in [2].

In [8], we derived the exact expressions of the first and second moments. Rewriting the predictive mean $\mu(\mathbf{x}^*)$ as a linear combination of the covariance between the new $\mathbf{x}^*$ and the training points (as suggested in [12]), with our choice of covariance function, the calculation of $m(\mathbf{x}^*)$ then involves the product of two Gaussian functions:

$$m(\mu_{x^*}, \Sigma_{x^*}) = \int \mu(\mathbf{x}^*)p(\mathbf{x}^*)d\mathbf{x}^* = \sum_j \beta_j \int C(\mathbf{x}^*, \mathbf{x}_j)p(\mathbf{x}^*)d\mathbf{x}^* \tag{10}$$

with $\beta = \mathbf{K}^{-1}\ \mathbf{y}$. This leads to (refer to [9] for details)

$$m(\mu_{x^*}, \Sigma_{x^*}) = \mathbf{q^T}\beta \tag{11}$$

with $q_i = |\mathbf{W}^{-1}\mathbf{\Sigma}_{x^*} + I|^{-1/2}\exp\left(-\frac{1}{2}(\mu_{x^*} - \mathbf{x_i})^T(\mathbf{\Sigma}_{x^*} + \mathbf{W})^{-1}(\mu_{x^*} - \mathbf{x}_i)\right)$, where $\mathbf{W} = \mathrm{diag}[w_1^2, \ldots, w_D^2]$ and I is the $D \times D$ identity matrix.

In the same manner, we obtain for the variance

$$v(\mu_{x^*}, \Sigma_{x^*}) = C(\mu_{x^*}, \mu_{x^*} + \mathrm{Tr}\left[(\beta\beta^{\mathbf{T}} - \mathbf{K}^{-1})Q\right] - \mathrm{Tr}(\mathbf{q^T}\beta)^2 \tag{12}$$

with

$$\begin{aligned} Q_{ij} &= |2\mathbf{W}^{-1}\mathbf{\Sigma}_{x^*} + I|^{-1/2}\exp\left(-\frac{1}{2}(\mathbf{xb} - \mu_{x^*})^T(\frac{1}{2}\mathbf{W} + \mathbf{\Sigma}_{x^*})^{-1}(\mathbf{xb} - \mu_{x^*})\right) \\ &\quad \exp\left(-\frac{1}{2}(\mathbf{x_i} - \mathbf{x_j})^T(2\mathbf{W})^{-1}(\mathbf{x_i} - \mathbf{x_j})\right) \end{aligned} \tag{13}$$

where $\mathbf{xb} = (\mathbf{x}_i + \mathbf{x}_j)/2$.

3.2 Monte-Carlo alternative

Equation (5) can be solved by performing a numerical approximation of the integral, using a simple Monte-Carlo approach:

$$p(f(\mathbf{x}^*)|\mu_{x^*}, \mathbf{\Sigma}_{x^*}) = \int p(f(\mathbf{x}^*)|\mathbf{x}^*)p(\mathbf{x}^*)d\mathbf{x}^* \simeq \frac{1}{T}\sum_{t=1}^{T} p(f(\mathbf{x}^*)|\mathbf{x}^{*t}), \tag{14}$$

where $\mathbf{x}^{*t}$ are (independent) samples from $p(\mathbf{x}^*)$.

4 Iterative k-step ahead prediction of time series

For the multiple-step ahead prediction task of time series, the iterative method consists in making repeated one-step ahead predictions, up to the desired horizon. Consider the time series $y^{t_1}, \ldots, y^t$ and the state-space model $y^{t_i} = f(x^{t_i}) + \epsilon^{t_i}$ where $x^{t_i} = [y^{t_i-1}, \ldots, y^{t_i-L}]^T$ is the *state* at time t_i (we assume that the lag L is known) and the (white) noise has variance v_0.

Then, the"naive" iterative k-step ahead prediction method works as follows: it predicts only one time step ahead, using the estimate of the output of the current prediction, as well as previous outputs (up to the lag L), as the input to the prediction of the next time step, until the prediction k steps ahead is made. That way, only the output estimates are used and the uncertainty induced by each successive prediction is not accounted for.

Using the results derived in the previous section, we suggest to formally incorporate the uncertainty information about the intermediate regressor. That is, as we predict ahead in time, we now view the lagged outputs as random variables. In this framework, the input at time t_k is a random vector with mean formed by the predicted means of the lagged outputs $y^{t+k-\tau}$, $\tau = 1, \dots, L$, given by (11). The $L \times L$ input covariance matrix has the different predicted variances on its diagonal (with the estimated noise variance v_0 added to them), computed with (12), and the off-diagonal elements are given by, in the case of the exact solution, $\mathrm{cov}(y_{t_k}, x_{t_k}) = \sum_i \beta_i q_i (c_i - \mu_{x_{t_k}})$, where q_i is as defined previously and $c_i = C(\mathbf{W}^{-1}\mathbf{x}_i + \mathbf{\Sigma}_{x^*}^{-1}\mu_{x^*})$ with $C = (\mathbf{W}^{-1} + \mathbf{\Sigma}_{x^*}^{-1})^{-1}$.

4.1 Illustrative examples

The first example is intended to provide a basis for comparing the approximate and exact solutions, within the Gaussian approximation of (5)), to the numerical solution (Monte-Carlo sampling from the true distribution), when the uncertainty is propagated as we predict ahead in time. We use the second example, inspired from real-life problems, to show that iteratively predicting ahead in time without taking account of the uncertainties induced by each succesive prediction leads to inaccurate results, with unrealistically small error bars.

We then assess the predictive performance of the different methods by computing the average absolute error (L_1), the average squared error (L_2) and average minus log predictive density[2] (L_3), which measures the density of the actual true test output under the Gaussian predictive distribution and use its negative log as a measure of loss.

4.1.1 Forecasting the Mackey-Glass time series

The Mackey-Glass chaotic time series constitutes a wellknown benchmark and a challenge for the multiple-step ahead prediction task, due to its strong non-linearity [4]: $\frac{dz(t)}{dt} = -bz(t) + a\frac{z(t-\tau)}{1+z(t-\tau)^{10}}$. We have $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is re-sampled with period 1 and normalized. We choose $L = 16$ for the number of lagged outputs in the state vector, $\mathbf{x_k} = [\mathbf{z_{k-1}}, \mathbf{z_{k-2}}, \dots, \mathbf{z_{k-L}}]$ and the targets, $t_k = z_k$, are corrupted by a white noise with variance 0.001.

We train a GP model with a Gaussian kernel such as (2) on 100 points, taken at random from a series of 8000 points. Figure 1 shows the mean predictions with their uncertainties, given by the exact and approximate methods, and 50 samples from the Monte-Carlo numerical approximation, from $k = 1$ to $k = 100$ steps ahead, for different starting points. Figure 2 shows the plot of the 100-step ahead mean predictions (left) and their 2σ uncertainties (right), given by the exact and approximate methods, as well as the sample mean and sample variance obtained with the numerical solution (average over 50 points).

These figures show the better performance of the exact method on the approximate one. Also, they allow us to validate the Gaussian approximation, noticing that the error bars encompass the samples from the true distribution. Table 1 provides a quantitative confirmation.

Table 1: Average (over 500 test points) absolute error (L_1), squared error (L_2) and minus log predictive density (L_3) of the 100-step ahead predictions obtained using the exact method (M_1), the approximate one (M_2) and the sampling from the true distribution (M_3).

	L_1	L_2	L_3
M_1	0.4574	0.3397	0.8473
M_2	0.5409	0.4609	1.1300
M_3	0.4886	$.3765$	0.8979

[2] To evaluate these losses in the case of Monte-Carlo sampling, we use the sample mean and sample variance.

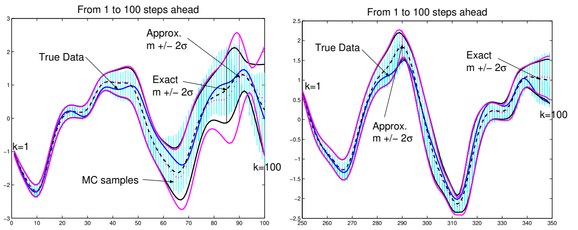

Figure 1: Iterative method in action: simulation from 1 to 100 steps ahead for different starting points in the test series. Mean predictions with 2σ error bars given by the exact (dash) and approximate (dot) methods. Also plotted, 50 samples obtained using the numerical approximation.

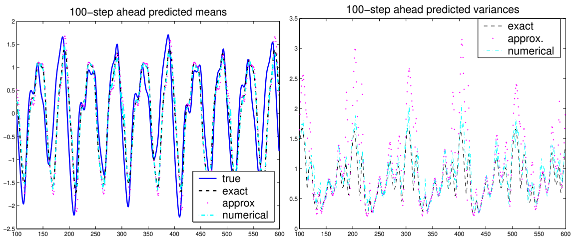

Figure 2: 100-step ahead mean predictions (left) and uncertainties (right.) obtained using the exact method (dash), the approximate (dot) and the sample mean and variance of the numerical solution (dash-dot).

4.1.2 Prediction of a pH process simulation

We now compare the iterative k-step ahead prediction results obtained when propagating the uncertainty (using the approximate method) and when using the output estimates only (the naive approach). For doing so, we use the pH neutralisation process benchmark presented in [3]. The training and test data consist of pH values (outputs y of the process) and a control input signal (u).

With a model of the form $y_t = f(y_{t-8}, \ldots, y_{t-1}, u_{t-8}, \ldots, u_{t-1})$, we train our GP on 1228 examples and consider a test set of ?? points (all data have been normalized).

Figure 3 shows the 10-step ahead predicted means and variances obtained when propagating the uncertainty and when using information on the past predicted means only. The losses calculated are the following: $L_1 = 0.1716$, $L_2 = 0.0574$ and $L_3 = 0.6208$ for the approximate method and $L_3 = 1980.2$ for the naive one!

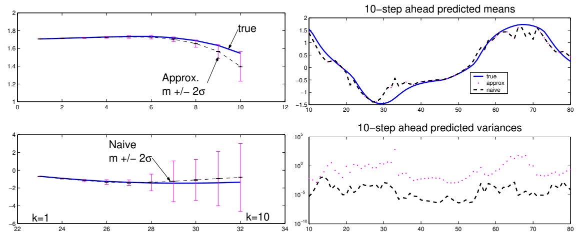

Figure 3: Predictions from 1 to 10 steps ahead (left). 10-step ahead mean predictions with the corresponding variances, when propagating the uncertainty (dot) and when using the previous point estimates only (dash).

5 Conclusions

We have presented a novel approach which allows us to use knowledge of the variance on inputs to Gaussian process models to achieve more realistic prediction variance in the case of noisy inputs.

Iterating this approach allows us to use it as a method for efficient propagation of uncertainty in the multi-step ahead prediction task of non-linear time-series. In experiments on simulated dynamic systems, comparing our Gaussian approximation to Monte Carlo simulations, we found that the propagation method is comparable to Monte Carlo simulations, and that both approaches achieved more realistic error bars than a naive approach which ignores the uncertainty on current state.

This method can help understanding the underlying dynamics of a system, as well as being useful, for instance, in a model predictive control framework where knowledge of the accuracy of the model predictions over the whole prediction horizon is required (see [6] for a model predictive control law based on Gaussian processes taking account of the prediction uncertainty). Note that this method is also useful in its own right in the case of noisy model inputs, assuming they have a Gaussian distribution.

Acknowledgements

Many thanks to Mike Titterington for his useful comments. The authors gratefully acknowledge the support of the *Multi-Agent Control* Research Training Network - EC TMR grant HPRN-CT-1999-00107 and RM-S is grateful for EPSRC grant *Modern statistical approaches to off-equilibrium modelling for nonlinear system control* GR/M76379/01.

References

[1] O'Hagan, A. (1978) Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society B* **40**:1-42.

[2] Girard, A. & Rasmussen, C. E. & Murray-Smith, R. (2002) Gaussian Process Priors With Uncertain Inputs: Multiple-Step Ahead Prediction. Technical Report, TR-2002-119, Dept. of Computing Science, University of Glasgow.

[3] Henson, M. A. & Seborg, D. E. (1994) Adaptive nonlinear control of a pH neutralisation process. *IEEE Trans Control System Technology* **2**:169-183.

[4] Mackey, M. C. & Glass, L. (1977) Oscillation and Chaos in Physiological Control Systems. *Science* **197**:287-289.

[5] MacKay, D. J. C. (1997) Gaussian Processes - A Replacement for Supervised Neural Networks?. Lecture notes for a tutorial at NIPS 1997.

[6] Murray-Smith, R. & Sbarbaro-Hofer, D. (2002) Nonlinear adaptive control using non-parametric Gaussian process prior models. *15th IFAC World Congress on Automatic Control, Barcelona*

[7] Neal, R. M. (1995) *Bayesian Learning for Neural Networks* PhD thesis, Dept. of Computer Science, University of Toronto.

[8] Quiñonero Candela, J & Girard, A. & Larsen, J. (2002) Propagation of Uncertainty in Bayesian Kernels Models – Application to Multiple-Step Ahead Forecasting *Submitted to ICASSP 2003*.

[9] Quiñonero Candela, J. & Girard, A. (2002) Prediction at an Uncertain Input for Gaussian Processes and Relevance Vector Machines - Application to Multiple-Step Ahead Time-Series Forecasting. Technical Report, IMM, Danish Technical University.

[10] Rasmussen, C. E. (1996) *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression* PhD thesis, Dept. of Computer Science, University of Toronto.

[11] Williams, C. K. I. & Rasmussen, C. E. (1996) Gaussian Processes for Regression *Advances in Neural Information Processing Systems 8* MIT Press.

[12] Williams, C. K. I. (2002) Gaussian Processes *To appear in The handbook of Brain Theory and Neural Networks, Second edition* MIT Press.

Appendix E

Propagation of Uncertainty in Bayesian Kernels Models - Application to Multiple-Step Ahead Forecasting

---

In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 701–704, Piscataway, New Jersey, IEEE. This paper was awarded an oral presentation.

In this paper we compare the approximate methods proposed in [D] with the exact moment matching described in Sect. 4.1.1.

PROPAGATION OF UNCERTAINTY IN BAYESIAN KERNEL MODELS — APPLICATION TO MULTIPLE-STEP AHEAD FORECASTING

Joaquin Quiñonero Candela(1), *Agathe Girard*(2), *Jan Larsen*(1) *& Carl Edward Rasmussen*(3)

(1)Informatics and Math. Modelling
Technical University of Denmark
Richard Petersens Plads, Build. 321
2800 Kongens Lyngby, Denmark
{jqc,jl}@imm.dtu.dk

(2)Dept. of Computing Science
Glasgow University
17 Lilybank Gardens
Glasgow G12 8QQ, Scotland
agathe@dcs.gla.ac.uk

(3)Biological Cybernetics
Max Planck Institute
Spemann straße 38
72076 Tübingen, Germany
carl@tuebingen.mpg.de

ABSTRACT

The object of Bayesian modelling is the predictive distribution, which in a forecasting scenario enables evaluation of forecasted values and their uncertainties. In this paper we focus on reliably estimating the predictive mean and variance of forecasted values using Bayesian kernel based models such as the Gaussian Process and the Relevance Vector Machine. We derive novel analytic expressions for the predictive mean and variance for Gaussian kernel shapes under the assumption of a Gaussian input distribution in the static case, and of a recursive Gaussian predictive density in iterative forecasting. The capability of the method is demonstrated for forecasting of time-series and compared to approximate methods.

1. INTRODUCTION

The problem of nonlinear forecasting is relevant to numerous application domains e.g. in financial modelling and control. This paper focuses on providing better estimates of the forecasted value as well as its uncertainty. The object of interest in Bayesian modelling framework [1] is the predictive density which contains all information about the forecasted value given the history of known values. For many Bayesian models the predictive density can only be approximated using Monte-Carlo sampling, local expansions, or variational approaches. However, when using Bayesian Gaussian shaped kernel models such as the Gaussian Process (GP) with a Gaussian kernel [1, 2] or the Relevance Vector Machine (RVM) [3, 4] the predictive mean and variance are given by analytic expressions under mild assumptions. Moreover the Bayesian kernel methods have proven to be very efficient nonlinear models [2, 4], with flexible approximation capabilities and high generalization performance.

We focus on the nonlinear auto-regressive (NAR) model with Gaussian innovations although more flexible nonlinear time-series models [5] sometimes are more efficient. Multi-step ahead forecasting can be done as direct forecast or as iterative one-step ahead forecasting. In [6] it is concluded that iterative forecasting usually is superior to direct forecasting. Generally the complexity of the nonlinear mapping in direct forecasting increases with the forecast horizon and for a fixed length time-series the number of training examples decreases with the forecast horizon. In iterative forecasting the complexity of the nonlinear mapping is much lower than in the direct case, the number of training samples higher, but the performance is diminished by the uncertainty of the forecasted values. Consequently the involved effects provide a delicate trade-off. We restrict this work to iterative forecasting, which offers the additional advantage that multi-step ahead forecasts can be obtained with only one trained model.

In classical iterative forecasting only the predictive mean is iterated, here we consider an improvement to the methods suggested in [7] which iterate both the predictive mean and variance. This corresponds to using the model in recall/test phase under uncertain input. We do not consider training the model under uncertain inputs, which has been addressed for nonlinear model in [8] and for linear models in e.g, [9].

In section 2 we introduce the Bayesian modelling framework. In section 3 we consider the evaluation of the prediction density with uncertain inputs, which is formulated for time-series forecasting in section 4. Finally section 5 provides numerical experiments, that demonstrate the capability of the proposed method.

2. BAYESIAN KERNEL MODELLING

Consider a D-dimensional column input vector $\boldsymbol{x}$ and a single output y, then the nonlinear model is defined as[1]

$$y = f(\boldsymbol{x}) + \varepsilon, \tag{1}$$

where $f(\cdot)$ is a nonlinear function implemented as a GP or a RVM, and $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is additive i.i.d. Gaussian noise with variance σ_ε^2. Suppose that the training data set is $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$, where N is the number of training samples. When using a GP [1, 2] or a RVM [3, 4], the predictive distribution of the output, y, is Gaussian [10],

$$p(y|\boldsymbol{x}, \mathcal{D}) = \mathcal{N}(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})), \tag{2}$$

where $\boldsymbol{x}$ is an arbitrary input at which we perform prediction. For a GP the mean and variance of the predictive distribution are given by

$$\mu(\boldsymbol{x}) = \boldsymbol{k}^\top(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{y}, \quad \sigma^2_{\mathrm{GP}}(\boldsymbol{x}) = 1 - \boldsymbol{k}^\top(\boldsymbol{x})\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{x}), \tag{3}$$

This work is supported by the Multi-Agent Control Research Training Network - EC TMR grant HPRN-CT-1999-00107. Roderick Murray-Smith is acknowledged for useful discussions.

[1] We tacitly assume that y has zero mean, although a bias term can be included, see further [10].

in *Proceedings of the 2003 IEEE Conference on Acoustics, Speech, and Signal Processing* (ICASSP03), Hong Kong, April 2003.

where $C_{\rm GP}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernel, which we set to the commonly used Gaussian form[2]. We have

$$
\begin{aligned}
C_{\rm GP}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp[-(\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{\Lambda}^{-1} (\boldsymbol{x}_i - \boldsymbol{x}_j)/2], && (4)\\
\boldsymbol{\Lambda} &= \mathrm{diag}[\lambda_1^2, \ldots, \lambda_D^2], && (5)\\
\boldsymbol{K} &= \{K_{ij}\} = \{C_{\rm GP}(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sigma_\varepsilon^2 \delta_{ij}\} && (6)\\
\boldsymbol{k}(\boldsymbol{x}) &= [C_{\rm GP}(\boldsymbol{x}, \boldsymbol{x}_1), \cdots, C_{\rm GP}(\boldsymbol{x}, \boldsymbol{x}_N)]^\top, && (7)\\
\boldsymbol{y} &= [y_1, \cdots, y_N]^\top. && (8)
\end{aligned}
$$

The kernel width hyper-parameters, λ_p, are fitted by maximizing the evidence (ML-II) using conjugate gradient, see e.g. [2].

For the RVM, let $\{\phi_j(\boldsymbol{x})\}$ and $\{\alpha_j\}$ with $j = 1, 2, \cdots, M$ be respectively the basis functions and the weight hyper-parameters, where M is the number of relevance vectors. Since typically $M < N$, the RVM yields sparse kernels, spanned by a finite number of basis functions [3, 10]. For the RVM the predictive distribution (2) has mean and variance specified by

$$\mu(\boldsymbol{x}) = \boldsymbol{\phi}^\top(\boldsymbol{x}) \boldsymbol{w}_{\rm MP}, \quad \sigma^2_{\rm RVM}(\boldsymbol{x}) = \boldsymbol{\phi}^\top(\boldsymbol{x}) \boldsymbol{\Sigma}^{-1} \boldsymbol{\phi}(\boldsymbol{x}), \qquad (9)$$

where, choosing Gaussian basis functions, we have

$$
\begin{aligned}
\boldsymbol{w}_{\rm MP} &= \sigma_\varepsilon^{-2} \boldsymbol{\Sigma} \boldsymbol{\Phi}^\top \boldsymbol{y}, && (10)\\
\boldsymbol{\Sigma} &= (\sigma_\varepsilon^{-2} \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \boldsymbol{A})^{-1}, && (11)\\
\boldsymbol{A} &= \mathrm{diag}[\alpha_1, \cdots, \alpha_M], && (12)\\
\phi_j(\boldsymbol{x}) &= \exp[-(\boldsymbol{x}_j - \boldsymbol{x})^\top \boldsymbol{\Lambda}^{-1} (\boldsymbol{x}_j - \boldsymbol{x})/2], && (13)\\
\boldsymbol{\phi}(\boldsymbol{x}) &= [\phi_1(\boldsymbol{x}), \cdots, \phi_M(\boldsymbol{x})]^\top, && (14)\\
\boldsymbol{\Phi} &= \{\Phi_{ij}\} = \{\phi_j(\boldsymbol{x}_i)\}, \; i = [1; N], \; j = [1; M]. && (15)
\end{aligned}
$$

The details of training the RVM are described in [3, 4].

3. PREDICTION WITH UNCERTAIN INPUT

Assume that the test input $\boldsymbol{x}$ can not be observed directly and the uncertainty is modeled as $\boldsymbol{x} \sim p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{u}, \boldsymbol{S})$, with mean $\boldsymbol{u}$ and covariance matrix $\boldsymbol{S}$. The resulting predictive distribution is then obtained by marginalizing over the test input

$$p(y|\boldsymbol{u}, \boldsymbol{S}, \mathcal{D}) = \int p(y|\boldsymbol{x}, \mathcal{D}) p(\boldsymbol{x})\, d\boldsymbol{x}. \qquad (16)$$

The principle is shown in Figure 1. The marginalization can in most cases only be carried out using Monte-Carlo numerical approximation techniques, however, in the case of Gaussian kernels[3] it is possible to obtain exact analytical expressions for the mean and variance of the marginalized predictive distribution:

$$
\begin{aligned}
m(\boldsymbol{u}, \boldsymbol{S}) &= \int y \cdot p(y|\boldsymbol{u}, \boldsymbol{S}, \mathcal{D})\, dy, \quad \text{and} && (17)\\
v(\boldsymbol{u}, \boldsymbol{S}) &= \int (y - m(\boldsymbol{u}, \boldsymbol{S}))^2 p(y|\boldsymbol{u}, \boldsymbol{S}, \mathcal{D})\, dy. && (18)
\end{aligned}
$$

The proposed method is an extension of the work presented in [7], which makes additional approximations, viz. Taylor series expansions of $\mu(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x})$ to first and second order around $\boldsymbol{u}$ and $\boldsymbol{S}$. Using properties of the conditional mean and variance

$$
\begin{aligned}
m(\boldsymbol{u}, \boldsymbol{S}) &= E_{\boldsymbol{x}}[E_y[y|\boldsymbol{x}]] = E_{\boldsymbol{x}}[\mu(\boldsymbol{x})], && (19)\\
v(\boldsymbol{u}, \boldsymbol{S}) &= E_{\boldsymbol{x}}[V_y[y|\boldsymbol{x}]] + V_{\boldsymbol{x}}[E_y[y|\boldsymbol{x}]]\\
&= E_{\boldsymbol{x}}[\sigma^2(\boldsymbol{x})] + V_{\boldsymbol{x}}[\mu(\boldsymbol{x})], && (20)
\end{aligned}
$$

where $E_{\boldsymbol{x}}[\cdot]$, $V_{\boldsymbol{x}}[\cdot]$ denote the expectation and variance wrt. $\boldsymbol{x}$. When using Gaussian kernels in GPs and Gaussian basis functions in RVMs, the expressions for $\mu(\boldsymbol{x})$ in eq. (3) and (9) are Gaussian shaped functions of $\boldsymbol{x}$ and the expressions for $\sigma^2(\boldsymbol{x})$ are products of Gaussian shaped functions in $\boldsymbol{x}$. Therefore the integrants involved in determining $m(\boldsymbol{u}, \boldsymbol{S})$ and $v(\boldsymbol{u}, \boldsymbol{S})$ are products of Gaussian shaped functions, which allows an analytical calculation. In [10] it is shown that

$$m(\boldsymbol{u}, \boldsymbol{S}) = \boldsymbol{\beta}^\top \boldsymbol{l}. \qquad (21)$$

For the GP $\boldsymbol{\beta} = \{\beta_1, \cdots, \beta_N\} = \boldsymbol{K}^{-1}\boldsymbol{y}$ and for the RVM $\boldsymbol{\beta} = \{\beta_1, \cdots, \beta_M\} = \boldsymbol{w}_{\rm MP}$. Vector $\boldsymbol{l} = \{l_1, \cdots, l_N\}$ is given by

$$
\begin{aligned}
l_j &= |\boldsymbol{\Lambda}^{-1}\boldsymbol{S} + \boldsymbol{I}|^{-\frac{1}{2}}\\
&\quad \cdot \exp\left[-\frac{1}{2}(\boldsymbol{u} - \boldsymbol{x}_j)^\top (\boldsymbol{\Lambda} + \boldsymbol{S})^{-1} (\boldsymbol{u} - \boldsymbol{x}_j)\right], \qquad (22)
\end{aligned}
$$

where $\boldsymbol{I}$ is the identity matrix. Note that if $\boldsymbol{S}$ is the zero matrix, then $\boldsymbol{l} = \boldsymbol{k}(\boldsymbol{u})$ and $m(\boldsymbol{u}, \boldsymbol{S}) = \mu(\boldsymbol{u})$ as would be expected.

Further, for the GP

$$
\begin{aligned}
v(\boldsymbol{u}, \boldsymbol{S}) &= \sigma^2_{\rm GP}(\boldsymbol{u}) + \mathrm{Tr}\left(\tilde{\boldsymbol{L}}(\boldsymbol{\beta}\boldsymbol{\beta}^\top - \boldsymbol{K}^{-1})\right)\\
&\quad + \mathrm{Tr}\left((\boldsymbol{k}(\boldsymbol{u})\boldsymbol{k}(\boldsymbol{u})^\top - \boldsymbol{l}\boldsymbol{l}^\top)\boldsymbol{\beta}\boldsymbol{\beta}^\top\right), \qquad (23)
\end{aligned}
$$

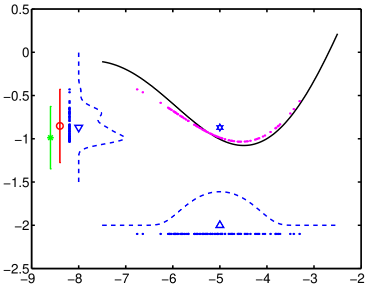

Fig. 1. Prediction with uncertain input. *On the x-axis*, the dashed line represents the Gaussian input distribution, with mean located by the triangle, from which we draw 100 samples (dots under it). *In the middle of the figure*, the solid line represents the true underlying function. We fit a model to it, and propagate the 100 input samples through the model (dots close to the true function). *On the y-axis* we project the 100 predicted values (dots) and use them to estimate the predictive density (dashed line), with mean located by the triangle. The error bar with a circle and the error bar with a star show the mean and 95% confidence interval of the Gaussian approximation with exact computation of mean and variance and of the method with Taylor expansion respectively.

[2]The exponential in equation (4) is usually multiplied by an additional hyperparameter whose value is fitted during training. We here set it to 1 for clarity, which requires normalizing the data to unit variance.

[3]Exact analytical results can also be obtained for polynomial kernels, $C(\boldsymbol{x}_p, \boldsymbol{x}_q) \propto |\boldsymbol{x}_p - \boldsymbol{x}_q|^r$, e.g. a linear model.

where $\tilde{\boldsymbol{L}} = \boldsymbol{L} - \boldsymbol{k}(\boldsymbol{u})\boldsymbol{k}(\boldsymbol{u})^\top$ and the elements of matrix $\boldsymbol{L}$ are

$$L_{ij} = \boldsymbol{k}_i(\boldsymbol{u})\boldsymbol{k}_j(\boldsymbol{u}) \cdot |2\boldsymbol{\Lambda}^{-1}\boldsymbol{S} + \boldsymbol{I}|^{-\frac{1}{2}} \quad (24)$$
$$\cdot \exp\Big[2(\boldsymbol{u} - \boldsymbol{x}_d)^\top \boldsymbol{\Lambda}^{-1}(2\boldsymbol{\Lambda}^{-1} + \boldsymbol{S}^{-1})^{-1}\boldsymbol{\Lambda}^{-1}(\boldsymbol{u} - \boldsymbol{x}_d)\Big],$$

and where $\boldsymbol{x}_d = \frac{1}{2}(\boldsymbol{x}_i + \boldsymbol{x}_j)$. For the RVM

$$v(\boldsymbol{u}, \boldsymbol{S}) = \sigma^2_{\mathrm{RVM}}(\boldsymbol{u}) + \mathrm{Tr}\left(\tilde{\boldsymbol{L}}(\boldsymbol{\beta}\boldsymbol{\beta}^\top + \boldsymbol{\Sigma}^{-1})\right) + \mathrm{Tr}\left([\boldsymbol{k}(\boldsymbol{u})\boldsymbol{k}(\boldsymbol{u})^\top - \boldsymbol{l}\boldsymbol{l}^\top]\boldsymbol{\beta}\boldsymbol{\beta}^\top\right). \quad (25)$$

Notice that both for GPs and RVMs, when $\boldsymbol{S}$ is the zero matrix, $\tilde{\boldsymbol{L}}$ is also the zero matrix, again $\boldsymbol{l} = \boldsymbol{k}(\boldsymbol{u})$, and $v(\boldsymbol{u}, \boldsymbol{S}) = \sigma^2(\boldsymbol{u})$.

4. APPLICATION TO TIME-SERIES FORECASTING

Suppose that $\{y_t\}$ are the ordered samples of a time-series, where t is an integer time index. We wish to make time-series forecasting using a NAR model (1), where the inputs are formed by a collection of previous output values, $\boldsymbol{x}_t = [y_{t-1}, y_{t-2}, \ldots, y_{t-L}]$, where the integer L is the size of the lag space.

Given that we have observed the values $y^T \equiv \{y_t\}_{t=1}^T$, T being the number of observed samples, computing the predictive density of the value y_{T+1} is readily given by the model from (2) as

$$p(y_{T+1}|\boldsymbol{x}_{T+1}) = \mathcal{N}(\mu(\boldsymbol{x}_{T+1}), \sigma^2(\boldsymbol{x}_{T+1}))$$

The predictive density of the value y_{T+2} (two steps ahead) depends on $\boldsymbol{x}_{T+1}$, which now contains a stochastic element. In general, the predictive distribution of y_{T+k}, with $k \geq 2$, requires integrating out the uncertainty of the input:

$$p(y_{T+k}|y^T) = \int p(y_{T+k}|\boldsymbol{x}_{T+k})\, p(\boldsymbol{x}_{T+k}|y^T)\, d\boldsymbol{x}_{T+k}. \quad (26)$$

It is straightforward that this scheme leads to a recursive density estimation. The integral in (26) has no analytical solution. A naïve approach to the recursion is to ignore the uncertainty in the distribution of the input by setting $p(\boldsymbol{x}_{T+k}) = \delta(\boldsymbol{x} - [\mu(\boldsymbol{x}_{T+k-1}), \ldots, \mu(\boldsymbol{x}_{T+k-L})]^\top)$[4], thus propagating only the mean predictions. This method yields very poor error-bars, since it in some way only considers one step ahead predictions, treating the previous predicted values as exact, and is therefore overconfident, [7]. Alternatively, one can approximate the predictive density of y_{T+k} by a Gaussian density and compute only the mean and variance of $p(y_{T+k}|y^T)$. By doing this one ensures that the input distribution $p(\boldsymbol{x}_{T+k}|y^T)$ is always Gaussian, which allows to use the results described in section 3 for computing the mean and variance of y_{T+K}, see eq. (26). This can be done exactly (for Gaussian or polynomial kernels) or in an approximate fashion, [7]. The recursive mechanism works because the predictive distribution of y_{T+1} at the first step is Gaussian (26), and therefore the input distribution of $\boldsymbol{x}_{T+1}$ is also Gaussian. We call this procedure of recursively approximating the predictive density by a Gaussian the Recursive Gaussian Predictive Density (RGPD), and distinguish between exact-RGPD for the case of exact computation of mean and variance and approximate-RGPD for the case where the model is approximated by a Taylor expansion, [7].

[4] Where $\delta(x)$ is 1 for $x = 0$ and 0 otherwise. If $k < L$, we have simply $\mu(\boldsymbol{x}_n) = y_n$ for $n < T$.

In the RGPD scheme, the input distribution is given by[5]

$$p(\boldsymbol{x}_{T+k}|y^T) = \mathcal{N}(\mathbf{u}_{T+k}, \mathbf{S}_{T+k}), \quad (27)$$

where

$$\mathbf{u}_{T+k} = [m(\boldsymbol{u}_{T+k-1}, \boldsymbol{S}_{T+k-1}), \ldots, m(\boldsymbol{u}_{T+k-L}, \boldsymbol{S}_{T+k-L})],$$

and where $\mathbf{S}_{T+k}$ is iteratively computed by using the fact that its first column is given by

$$(\mathbf{S}_{T+k})_{1:L,1} = \mathrm{cov}(y_{T+k}, \boldsymbol{x}_{T+k}) = \sum_j \beta_j L_j(\mathbf{c}_j - \mathbf{u}_{T+k}), \quad (28)$$

where $\mathbf{c}_j = (\boldsymbol{\Lambda}^{-1} + \boldsymbol{S}^{-1})^{-1}(\boldsymbol{\Lambda}^{-1}\boldsymbol{x}_j + \boldsymbol{S}^{-1}\boldsymbol{u})$, refer to [10].

5. EXPERIMENTS

We examine the comparative performance of the exact and approximate-RGPD on a hard prediction problem, the Mackey-Glass chaotic time series [11], which is well-known for its strong non-linearity. In [4] we showed that non-linear models, in particular RVMs, have a prediction error four orders of magnitude lower than optimized linear models. The Mackey-Glass attractor is a non-linear chaotic system described by the following equation:

$$\frac{dz(t)}{dt} = -bz(t) + a\frac{z(t-\tau)}{1 + z(t-\tau)^{10}} \quad (29)$$

where the constants are set to $a = 0.2$, $b = 0.1$ and $\tau = 17$. The series is re-sampled with period 1 according to standard practice. The inputs are formed by $L = 16$ samples spaced 1 periods from each other $\mathbf{x}_k = [z_{k-1}, z_{k-2}, \ldots, z_{k-L}]$ and the targets are chosen to be $y_k = z_k$.

We train a GP model with Gaussian kernel on only 100 examples — enough to obtain a 1-step ahead normalized mean squared error on the order of 10^{-4}. Besides, we normalize the data and contaminate it with a small amount of Gaussian noise with variance 10^{-3}. Figure 2 shows the result of making 100 iterative predictions using a GP model, both for the exact-RGPD and the approximate-RGPD methods. By informal visual inspection, the error-bars of the exact-RGPD seem to be better than those of the approximate-RGPD. Consequently the exact-RGPD produces a better predictive density, which we show in figure 3. The mean value of the predictions seems also to be a slightly closer to the true target values for the exact-RGPD than for the approximate-RGPD.

In order to better evaluate the performance of the proposed methods, for a given prediction horizon, we compute the negative log predictive density, the squared error and the absolute error. While the two last measures only take into consideration the mean of the Gaussian predictive distribution, the first one also takes into account its variance. We average over 200 repetitions with different starting points (chosen at random from the series), and represent averages of the three loss measures for prediction horizons ranging from 1 to 100. Figure 3 shows the results. The means are slightly better for the exact-RGPD, but the predictive distribution is much improved. The better error-bars obtained by the exact-RGPD result in a lower value of the negative log predictive density for all values of the prediction horizon. The performance of the naïve iterative method is identical to that of the approximate-RGPD in terms of absolute and squared error. In terms of predictive density (since it produces unrealistic small error-bars) its performance is so poor that it is not worth reporting.

[5] If $k < L$, we have simply $\boldsymbol{\mu}(\boldsymbol{x}_n) = y_n$ for $n < T$.

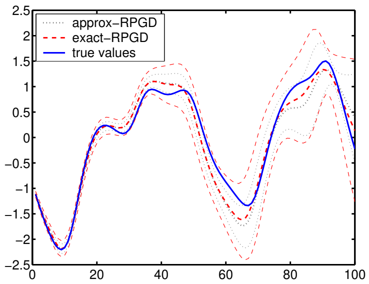

Fig. 2. 100 iterated predictions for the exact-RGPD (dashed) and approximate-RGPD (dotted): for each the thicker lines represent the mean of the predictive distributions and the two thinner lines around represent the upper and lower bounds of the 95% confidence interval of the Gaussian predictive distributions. The solid line shows the true target values.

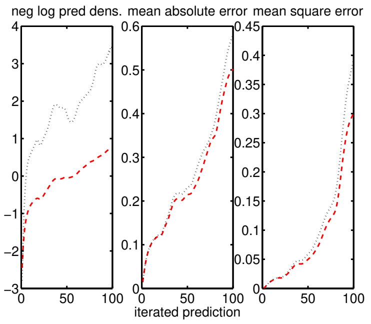

Fig. 3. Negative log predictive density, mean absolute error and mean squared error as a function of the iterative prediction horizon for the exact-RGPD method (dashed) and for the approximate-RGPD (dotted). Averages over 200 repetitions.

6. CONCLUSIONS

We have derived analytical expressions for the exact computation of the mean and variance of the marginalized predictive distribution for uncertain Gaussian test inputs. This analytical expressions are valid for Gaussian processes and the Relevance Vector Machine (extended linear models) with Gaussian or polynomial kernels or basis functions. Our results extend the approximate method presented in [7], where the mean prediction was unaffected by the input uncertainty. In our case the input uncertainty biases the mean prediction, by smoothing, which is interesting in itself for predictions on noisy inputs. Furthermore, in the context of iterated time-series forecasting, our exact-RGPD not only gives much better error-bars, but the mean predictions are closer to the true values, both in terms of absolute and squared error. We are currently investigating efficient Monte Carlo methods to avoid the Gaussian approximation of the recursive predictive density.

7. REFERENCES

[1] Radford M. Neal, *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics, no. 118. Springer, New York, 1996.

[2] Carl E. Rasmussen, *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*, Ph.D. thesis, Dept. of Computer Science, University of Toronto, 1996.

[3] Michael E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[4] Joaquin Quiñonero-Candela and Lars Kai Hansen, "Time series prediction based on the relevance vector machine with adaptive kernels," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002, pp. 985–988.

[5] I.J. Leontaritis and S.A. Billings, "Input-output parametric models for non-linear systems, part 1: Deterministic non-linear systems, part 2: Stochastic non-linear systems," *International Journal of Control*, vol. 41, pp. 303–344, 1985.

[6] J. Doyne Farmer and John J. Sidorowich, "Exploiting chaos to predict the future and reduce noise," Tech. Rep. LA-UR-88, Los Alamos National Laboratory, 1988.

[7] Agathe Girard, Carl Edward Rasmussen, and Roderick Murray-Smith, "Gaussian process with uncertain input - application to multiple-step ahead time-series forecasting," in *Advances in Neural Information Processing Systems 15*, Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, Eds. 2003, MIT Press.

[8] Volker Tresp, Subutai Ahmad, and Ralph Neuneier, "Training neural networks with deficient data," in *Advances in Neural Information Processing Systems 6*, Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, Eds. 1994, pp. 128–135, Morgan Kaufmann Publishers, Inc.

[9] Oscar Nestares, David J. Fleet, and David J. Heeger, "Likelihood functions and confidence bounds for total-least-squares problems," in *Proceedings of IEEE Conference on Computer Vision and Pattern Regcognition*, 2000, vol. 1, pp. 523–530.

[10] Joaquin Quiñonero-Candela and Agathe Girard, "Prediction at an uncertain input for gaussian processes and relevance vector machines - application to multiple-step ahead time-series forecasting," Tech. Rep., IMM, DTU, 2002, http://isp.imm.dtu.dk/staff/jqc/prop_uncert.ps.gz.

[11] M.C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, July 1977.

Appendix F

Learning Depth from Stereo

To appear in *Deutsche Arbeitsgemeinschaft für Mustererkennung (DAGM) Pattern Recognition Symposium 26*, Heidelberg, Germany. Springer.

In this paper we apply Gaussian Process models to a real world task: 3D stereo vision. While traditionally the task of inferring the spatial position of an object from the local coordinates in two cameras has been addressed with classic camera calibration techniques, and photogrammetric models, in this paper we view it as a "black box" machine learning regression task. Gaussian Processes excel at this task, compared to other machine learning methods.

Learning Depth From Stereo

Fabian H. Sinz[1], Joaquin Quiñonero Candela[2], Gökhan H. Bakır[1], Carl E. Rasmussen[1], and Matthias O. Franz[1]

[1] Max Planck Institute for Biological Cybernetics
Spemannstraße 38, 72076 Tübingen
{fabee;jqc;gb;carl;mof}@tuebingen.mpg.de
[2] Informatics and Mathematical Modelling, Technical University of Denmark,
Richard Petersens Plads, B321, 2800 Kongens Lyngby, Denmark
jqc@imm.dtu.dk

Abstract. We compare two approaches to the problem of estimating the depth of a point in space from observing its image position in two different cameras: 1. The classical photogrammetric approach explicitly models the two cameras and estimates their intrinsic and extrinsic parameters using a tedious calibration procedure; 2. A generic machine learning approach where the mapping from image to spatial coordinates is directly approximated by a *Gaussian Process* regression. Our results show that the generic learning approach, in addition to simplifying the procedure of calibration, can lead to higher depth accuracies than classical calibration although no specific domain knowledge is used.

1 Introduction

Inferring the three-dimensional structure of a scene from a pair of stereo images is one of the principal problems in computer vision. The position $\mathbf{X} = (X, Y, Z)$ of a point in space is related to its image at $\mathbf{x} = (x, y)$ by the equations of perspective projection

$$x = x_0 - s_{xy}c \cdot \frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} + \Xi_x(\mathbf{x}) \tag{1}$$

$$y = y_0 - c \cdot \frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} + \Xi_y(\mathbf{x}) \tag{2}$$

where $\mathbf{x}_0 = (x_0, y_0)$ denotes the image coordinates of the principal point of the camera, c the focal length, $\mathbf{X}_0 = (X_0, Y_0, Z_0)$ the 3D-position of the camera's optical center with respect to the reference frame, and r_{ij} the coefficients of a 3×3 rotation matrix R describing the orientation of the camera. The factor s_{xy} accounts for the difference in pixel width and height of the images, the 2-D-vector field $\Xi(\mathbf{x})$ for the lens distortions.

The classical approach to stereo vision requires a *calibration procedure* before the projection equations can be inverted to obtain spatial position, i.e., estimating the *extrinsic* ($\mathbf{X}_0$ and R) and *intrinsic* ($\mathbf{x}_0$, c, s_{xy} and Ξ) parameters of each camera from a set of points with known spatial position and their corresponding image positions. This is normally done by repeatedly linearizing the projection equations and applying a standard least square estimator to obtain an iteratively refined estimate of the camera parameters [1]. This approach neglects the nonlinear nature of the problem, which causes that its convergence critically depends on the choice of the initial values for the parameters. Moreover, the right choice of the initial values and the proper setup of the models can be a tedious procedure.

The presence of observations and desired target values on the other hand, makes depth estimation suitable for the application of nonlinear supervised learning algorithms such as *Gaussian Process Regression*. This algorithm does not require any specific domain knowledge and provides a direct solution to nonlinear estimation problems. Here, we investigate whether such a machine learning approach can reach a comparable performance to classical camera calibration. This can lead to a considerable simplification in practical depth estimation problems as off-the-shelf algorithms can be used without specific adaptations to the setup of the stereo problem at hand.

2 Classical Camera Calibration

As described above, the image coordinates of a point are related to the cameras parameters and its spatial position by a nonlinear function $\mathbf{F}$ (see Eqs. 1 and 2)

$$\mathbf{x} = \mathbf{F}(\mathbf{x}_0, c, s_{xy}, R, \mathbf{X}_0, \Xi, \mathbf{X}) \tag{3}$$

The estimation of parameters is done by a procedure called *bundle adjustment* which consists of iteratively linearizing the camera model in parameter space and estimating an improvement for the parameter from the error on a set of m known pairs of image coordinates $\mathbf{x}_i = (x_i, y_i)$ and spatial coordinates $\mathbf{X}_i = (X_i, Y_i, Z_i)$. These can be obtained from an object with a distinct number of points whose coordinates with respect to some reference frame are known with high precision such as, for instance, a calibration rig.

Before this can be done, we need to choose a low-dimensional parameterization of the lens distortion field Ξ because otherwise the equation system 3 for the points $1 \ldots m$ would be underdetermined. Here, we model the x- and y-component of Ξ as a weighted sum over products of one-dimensional Chebychev polynomials T_i in x and y, where i indicates the degree of the polynomial

$$\Xi_x(\mathbf{x}) = \sum_{i,j=0}^{t} a_{ij} T_i(s_x x) T_j(s_y y), \; \Xi_y(\mathbf{x}) = \sum_{i,j=0}^{t} b_{ij} T_i(s_x x) T_j(s_y y), \tag{4}$$

The factors s_x, s_y scale the image coordinates to the Chebychev polynomials' domain $[-1, 1]$. In the following, we denote the vector of the complete set of camera parameters by $\theta = (\mathbf{x}_0, c, s_{xy}, R, \mathbf{X}_0, a_{11}, \ldots, a_{tt}, b_{11}, \ldots, b_{tt})$.

In the iterative bundle adjustment procedure, we assume we have a parameter estimate θ_{n-1} from the previous iteration. The residual $\mathbf{l}_i$ of point i for the camera model from the previous iteration is then given by

$$\mathbf{l}_i = \mathbf{x}_i - \mathbf{F}(\theta_{n-1}, \mathbf{X}_i). \tag{5}$$

This equation system is linearized by computing the Jacobian $\mathcal{J}(\theta_{n-1})$ of $\mathbf{F}$ at θ_{n-1} such that we obtain

$$\mathbf{l} \approx \mathcal{J}(\theta_{n-1})\Delta\theta \tag{6}$$

where $\mathbf{l}$ is the concatenation of all $\mathbf{l}_i$ and $\Delta\theta$ is the estimation error in θ that causes the residuals. Usually, one assumes a prior covariance Σ_{ll} on $\mathbf{l}$ describing the inaccuracies in the image position measurements. $\Delta\theta$ is then obtained from a standard linear estimator [3]

$$\Delta\theta = (\mathcal{J}^\top \Sigma_{ll}^{-1} \mathcal{J})^{-1} \mathcal{J} \Sigma_{ll}^{-1} \mathbf{l}\ . \tag{7}$$

Finally, the new parameter estimate θ_n for iteration n is improved according to $\theta_n = \theta_{n-1} + \Delta\theta$. Bundle adjustment needs a good initial estimate θ_0 for the camera parameters in order to ensure that the iterations converge to the correct solution. There exists a great variety of procedures for obtaining initial estimates which have to be specifically chosen for the application (e.g. aerial or near-range photogrammetry).

The quality of the estimation can still be improved by modelling uncertainties in the spatial observations $\mathbf{X}_i$. This can be done by including all spatial observations in the parameter set and updating them in the same manner which requires the additional choice of the covariance Σ_{XX} of the measurements of spatial position [1]. Σ_{XX} regulates the tradeoff between the trust in the accuracy of the image observations on the one hand and the spatial observations on the other hand. For more detailed information on bundle adjustment please refer to [1].

Once the parameter sets $\theta^{(1)}$ and $\theta^{(2)}$ of the two camera models are known, the spatial position $\mathbf{X}^*$ of a newly observed image point ($\mathbf{x}_1^*$ in the first and $\mathbf{x}_2^*$ in the second camera) can be estimated using the same technique. Again, $\mathbf{F}$ describes the stereo camera's mapping from spatial to image coordinates according to Eqns. 1 and 2

$$\mathbf{x}_k^* = \mathbf{F}(\theta^{(k)}, \mathbf{X}^*), \quad k = 1, 2 \tag{8}$$

but this time the θ are kept fixed and the bundle adjustment is computed for estimates of $\mathbf{X}^*$ [1].

3 Gaussian Process Regression

The machine learning algorithm used in our study assumes that the data are generated by a Gaussian Process (GP). Let us call $f(\mathbf{x})$ the non-linear function that maps the D-dimensional input $\mathbf{x}$ to a 1-dimensional output. Given an arbitrary set of inputs $\{\mathbf{x}_i | i = 1, \ldots, m\}$, the joint prior distribution of the corresponding function evaluations $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m)]^\top$ is jointly Gaussian:

$$p(\mathbf{f} | \mathbf{x}_1, \ldots, \mathbf{x}_m, \theta) \sim \mathcal{N}(0, K)\ , \tag{9}$$

with zero mean (a common and arbitrary choice) and covariance matrix K. The elements of K are computed from a parameterized covariance function, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j, \theta)$, where θ now represents the GP parameters. In Sect. 4 we present the two covariance functions we used in our experiments.

We assume that the output observations y_i differ from the corresponding function evaluations $f(\mathbf{x}_i)$ by Gaussian additive i.i.d. noise of mean zero and variance σ^2. For simplicity in the notation, we absorb σ^2 in the set of parameters θ. Consider now that we have observed the targets $\mathbf{y} = [y_1, \ldots, y_m]$ associated to our arbitrary set of m inputs, and would like to infer the predictive distribution of the unknown target y_* associated to a new input $\mathbf{x}_*$. First we write the joint distribution of all targets considered, easily obtained from the definition of the prior and of the noise model:

$$p\left(\begin{bmatrix}\mathbf{y}\\ y_*\end{bmatrix}\middle|\mathbf{x}_1,\ldots,\mathbf{x}_m,\theta\right) \sim \mathcal{N}\left(0, \begin{bmatrix}K+\sigma^2\mathcal{I} & \mathbf{k}_*\\ \mathbf{k}_*^\top & k(\mathbf{x}_*,\mathbf{x}_*)+\sigma^2\end{bmatrix}\right) \ , \tag{10}$$

where $\mathbf{k}_* = [k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_m)]^\top$ is the covariance between y_* and $\mathbf{y}$, and $\mathcal{I}$ is the identity matrix. The predictive distribution is then obtained by conditioning on the observed outputs $\mathbf{y}$. It is Gaussian:

$$p(y_*|\mathbf{y}, \mathbf{x}_1, \ldots, \mathbf{x}_m, \theta) \sim \mathcal{N}(m(\mathbf{x}_*), v(\mathbf{x}_*)) \ , \tag{11}$$

with mean and variance given respectively by:

$$\begin{aligned} m(\mathbf{x}_*) &= \mathbf{k}_*^\top[K+\sigma^2\mathcal{I}]^{-1}\mathbf{y} \ ,\\ v(\mathbf{x}_*) &= \sigma^2 + k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top[K+\sigma^2\mathcal{I}]^{-1}\mathbf{k}_* \ . \end{aligned} \tag{12}$$

Given our assumptions about the noise, the mean of the predictive distribution of $f(\mathbf{x}_*)$ is also equal to $m(\mathbf{x}_*)$, and it is the optimal point estimate of $f(\mathbf{x}_*)$. It is interesting to notice that the prediction equation given by $m(\mathbf{x}_*)$ is identical to the one used in Kernel Ridge Regression (KRR) [2]. However, GPs differ from KRR in that they provide full predictive distributions.

One way of learning the parameters θ of the GP is by maximizing the evidence of the observed targets $\mathbf{y}$ (or marginal likelihood of the parameters θ). In practice, we equivalently minimize the negative log evidence, given by:

$$-\log p(\mathbf{y}|\mathbf{x}_1, \ldots, \mathbf{x}_1, \theta) = \frac{1}{2}\log|K+\sigma^2\mathcal{I}| + \frac{1}{2}\mathbf{y}^\top[K+\sigma^2\mathcal{I}]^{-1}\mathbf{y} \ . \tag{13}$$

Minimization is achieved by taking derivatives and using conjugate gradients. An alternative way of inferring θ is to use a Bayesian variant of the leave-one-out error (GPP, Geisser's surrogate predictive probability, [4]). In our study we will use both methods, choosing the most appropriate one for each of our two covariance functions. More details are provided in Sect. 4.

4 Experiments

Dataset. We used a robot manipulator holding a calibration target with a flattened LED to record the data items. The target was moved in planes of different

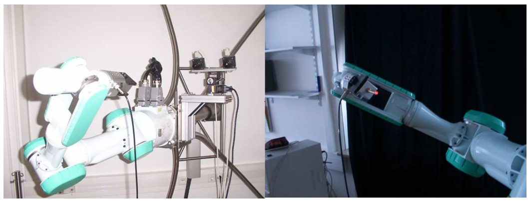

Fig. 1. Robot arm and calibration target, which were used to record the data items.

depths, perpendicular to the axis of the stereo setup. The spatial position of the LED was determined from the position encoders of the robot arm with a nominal positioning accuracy of $0.01mm$. The center of the LED was detected using several image processing steps. First, a threshold operation using the upper 0.01 percentile of the image's gray-scale values predetected the LED. Then a two-dimensional spline was fitted through a window around the image of the LED with an approximate size of $20px$. A *Sobel* operator was used as edge detector on the spline and a *Zhou* operator located the LED center with high accuracy (see [1]). We recorded 992 pairs of spatial and image positions, 200 of which were randomly selected as training set. The remaining 792 were used as test set.

Classical calibration. During bundle adjustment, several camera parameters were highly correlated with others. Small variations of these parameters produced nearly the same variation of the function values of $\mathbf{F}$, which lead to a linear dependency of the columns of $\mathcal{J}$ and thus to a rank deficiency of $\mathcal{J}^\top \Sigma_{ll}^{-1} \mathcal{J}$. Therefore, the parameters of a correlating pair could not be determined properly. The usual way to deal with this problem is to exclude one of the correlating parameters from the estimation. As both the principal point $\mathbf{x}_0$ and the coefficients a_{00}, b_{00} highly correlated with camera yaw and pitch, we assumed them to be zero and excluded them from estimation. Furthermore a_{10}, b_{01}, a_{12} and b_{21} were excluded because of their correlation with s_{xy} and c. As the combination $a_{01} = -b_{10}$ showed a high correlation with the roll angle of the camera, the parameter a_{01} was not estimated and its value was set to b_{01}. The correlations of a_{20} and a_{02} with camera yaw resp. b_{20} and b_{02} with camera pitch were removed by setting a_{20} to b_{02} and a_{02} to b_{20}. Higher degree polynomials in the parameterization of the lens distortion field induce vector fields too complex to correlate with other parameters of the camera such that none had to be switched off due to correlations (see [6] for more detailed information on the parameterization of the camera model). We used a ten-fold crossvalidation scheme to determine whether the corresponding coefficients should be included in the model or not. The error in the image coordinates was assumed to be conditionally independent with $\sigma^2 = 0.25px$, so the covariance matrix Σ_{ll} became diagonal with $\Sigma_{ll} = 0.25 \cdot \mathcal{I}$.

The same assumption was made for Σ_{XX}, though the value of the diagonal elements was chosen by a ten fold cross validation.

Gaussian Process Regression. For the machine learning approach we used both the *inhomogeneous polynomial kernel*

$$k(x, x') = \sigma_\nu^2 \langle x, x' + 1 \rangle^g \tag{14}$$

of degree g and the *squared exponential kernel*

$$k(x, x') = \sigma_\nu^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{D} \frac{1}{\lambda_d^2} (x_d - x_d')^2 \right) . \tag{15}$$

with automatic relevance determination (ARD). Indeed, the lengthscales λ_d can grow to eliminate the contribution of any irrelevant input dimension.

The parameters σ_ν^2, σ^2 and g of the polynomial covariance function were estimated by maximizing the GPP criterion [4]. The parameters σ_ν^2, σ^2 and the λ_d of the squared exponential kernel were estimated by maximizing their marginal log likelihood [5]. In both cases, we used the conjugate gradient algorithm as optimization method.

We used two different types of preprocessing in the experiments: 1. Scaling each dimension of the input data to the interval $[-1, 1]$; 2. Transforming the input data according to

$$(x_1, y_1, x_2, y_2) \mapsto \left(\frac{1}{2}(x_1 - x_2), \frac{1}{2}(x_1 + x_2), \frac{1}{2}(y_1 - y_2), \frac{1}{2}(y_1 + y_2) \right) . \tag{16}$$

The output data was centered for training.

5 Results

The cross validation for the camera model yielded $\sigma_X = 2mm$ as best a priori estimation for the standard deviation of the spatial coordinates. In the same way, a maximal degree of $t = 3$ for the Chebychev polynomials was found to be optimal for the estimation of the lens distortion. Table 1 shows the test errors of the different algorithms and preprocessing methods.

All algorithms achieved error values under one millimeter. Gaussian Process regression with both kernels showed a superior performance to the classical approach. Fig. 5 shows the position error according to the test points actual depth and according to the image coordinates distance to the lens center, the so called *excentricity*. One can see that the depth error increases nonlinearly with increasing spatial distance to the camera. Calculation of errors shows that the depth error grows quadratically with the image position error, so this behaviour is expected and indicates the sanity of the learned model. Another hint that all of the used algorithms are able to model the lens distortions is the absence of a

Table 1. Test error for bundle adjustment and Gaussian Process Regression with various kernels, computed on a set of 792 data items. Root mean squared error of the spatial residua was used as error measure.

METHOD	TEST ERROR	PREPROCESSING
Bundle adjustment	0.38mm	-
Inhomogeneous polynomial	0.29mm	scaled input
Inhomogeneous polynomial	0.28mm	transformed, scaled input
Squared exponential	0.31mm	scaled input
Squared exponential	0.27mm	transformed, scaled input

trend in the right figure. Again, the learning algorithms do better and show a smaller error for almost all excentricities.

The superiority of the squared exponential kernel to the polynomial can be explained by its ability to assign different length scales to different dimensions of the data and therefore set higher weights on more important dimensions. In our experiments $\frac{1}{\lambda_1^2}$ and $\frac{1}{\lambda_3^2}$ were always approximately five times larger than $\frac{1}{\lambda_2^2}$ and $\frac{1}{\lambda_4^2}$, which is consistent with the underlying physical process, where the depth of a point is computed by the disparity in the x-direction of the image coordinates. The same phenomenon could be observed for the transformed inputs, where higher weights where assigned to the x_1 and x_2.

6 Discussion

We applied Gaussian Process Regression to the problem of estimating the spatial position of a point from its coordinates in two different images and compared its performance to the classical camera calibration. Our results show that the generic learning algorithms performed better although maximal physical knowledge was used in the explicit stereo camera modelling.

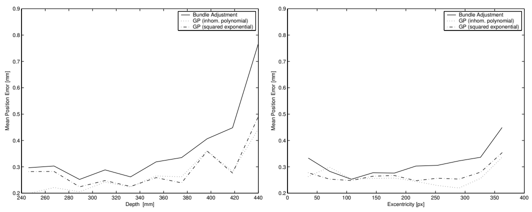

Fig. 2. Position error depending on the actual depth of the test point (left figure) and on the distance to the lens center, the so called *excentricity* (right figure).

An additional advantage of our approach is the mechanical and therefore simple way of model selection, while the correct parametrization of a camera model and elimination of correlating terms is a painful and tedious procedure. Moreover the convergence of the regression process does not depend on good starting values like the estimation of the camera model's parameters does.

A disadvantage of the machine learning approach is that it does not give meaningful parameters such as position and orientation in space or the camera's focal length. Moreover, it does not take into account situations where the exact spatial positions of the training examples are unknown, whereas classical camera calibration allows for an improvement of the spatial position in the training process.

The time complexity for all algorithms is $\mathcal{O}(m^3)$ for training and $\mathcal{O}(n)$ for the computation of the predictions, where m denotes the number of training examples and n the number of test examples. In both training procedures, matrices with a size in the order of the number of training examples have to be inverted at each iteration step. So the actual time needed also depends on the number of iteration steps, which scale with the number of parameters and can be assumed constant for this application. Without improving the spatial coordinates, the time complexity for the training of the camera model would be $\mathcal{O}(p^3)$, where p denotes the number of parameters. But since were are also updating the spatial observations, the number of parameters is upper bounded by a multiple of the number of training examples such that the matrix inversion in (7) is in $\mathcal{O}(m^3)$. An additional advantage of GP is the amount of time actually needed for computing the predictions. Although predicting new spatial points is in $\mathcal{O}(n)$ for GP and the camera model, predictions with the camera model always consume more time. This is due to the improvements of the initial prediction with a linear estimator which again is an iterative procedure involving an inversion of a matrix of constant size at each step.

References

1. Thomas Luhmann: Nahbereichsphotogrammetrie - Grundlagen, Methoden und Anwendungen. Wichmann (2000) [in German]
2. Nello Cristianini and John Shawe-Taylor: Support Vector Machines - and other kernel-based methods. Cambridge University Press (2000)
3. Steven M. Kay: Statistical Signal Processing Vol. I. Prentice Hall (1993)
4. S. Sundararajan, S.S. Keerthi: Predictive Approaches for Choosing Hyperparameters in Gaussian Processes. Neural Computation 13, 1103-1118 (2001). MIT
5. C. K. I. Williams and C. E. Rasmussen: Gaussian processes for regression. Advances in Neural Information Processing Systems 8 pp. 514-520 MIT Press (1996)
6. Fabian Sinz: Kamerakalibrierung und Tiefenschätzung - Ein Vergleich von klassischer Bündelblockausgleichung und statistischen Lernalgorithmen. http://www.kyb.tuebingen.mpg.de/∼fabee (2004) [in German]

Bibliography

Carroll, R. J., Ruppert, D., and Stefanski, L. A. (1995). *Measurement Error in Nonlinear Models*. Chapman and Hall, London.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley and Sons, Hoboken, New Jersey.

Csató, L. (2002). *Gaussian Processes – Iterative Sparse Approximation*. PhD thesis, Aston University, Birmingham, United Kingdom.

Csató, L. and Opper, M. (2002). Sparse online gaussian processes. *Neural Computation*, 14(3):641–669.

Dellaportas, P. and Stephens, D. A. (1995). Bayesian Analysis of Errors-in-Variables Regression Models. *Biometrics*, 51:1085–1095.

Dempster, N. M., Laird, A., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):185–197.

D'Souza, A., Vijayakumar, S., and Schaal, S. (2004). The bayesian backfitting relevance vector machine. In Dy, J., editor, *International Conference on Machine Learning 21*, San Francisco, California. Morgan Kaufmann Publishers.

Farmer, J. D. and Sidorowich, J. J. (1988). Exploiting chaos to predict the future and reduce noise. Technical Report LA-UR-88-901, Los Alamos National Laboratory, Los Alamos, New Mexico.

Faul, A. C. and Tipping, M. E. (2002). Analysis of sparse bayesian learning. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 383–389, Cambridge, Massachussetts. MIT Press.

Ghahramani, Z. and Jordan, M. I. (1994). Supervised learning from incomplete data via an EM approach. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 120–127, San Francisco, California. Morgan Kaufmann Publishers.

Gibbs, M. and MacKay, D. J. C. (1997). Efficient implementation of gaussian processes. Technical report, Cavendish Laboratory, Cambridge University, Cambridge, United Kingdom.

Girard, A. (2004). *Approximate Methods for Propagation of Uncertainty with Gaussian Process Models*. PhD thesis, Glasgow University, Glasgow, Scotland.

Girard, A., Rasmussen, C. E., and Murray-Smith, R. (2002). Gaussian process priors with uncertain inputs: Multiple-step ahead prediction. Technical report, Department of Computing Science, Glasgow University, Glasgow, Scotland.

Girard, A., Rasmussen, C. E., Quiñonero-Candela, J., and Murray-Smith, R. (2003). Gaussian process with uncertain inputs - application to multiple-step ahead time-series forecasting. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 529–536, Cambridge, Massachussetts. MIT Press.

Golub, G. H. and Loan, C. F. V. (1980). An analysis of the total least squares problem. *SIAM Journal of Numerical Analysis*, 17(6):883–893.

Harrison, D. and Rubinfeld, D. (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics & Management*, 5:81–102. Data available from http://lib.stat.cmu.edu/datasets/boston.

Hooke, R. and Jeeves, T. A. (1961). "Direct Search" solution of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2):212–229.

Jaynes, E. T. (2003). *Probability theory: the logic of science*. Cambridge Univesity Press, Cambridge, United Kingdom.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In Becker, S., Thrun, S., and Obermayer, K., editors, *Neural Information Processing Systems 15*, pages 609–616, Cambridge, Massachussetts. MIT Press.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

MacKay, D. J. C. (1994). Bayesian non-linear modelling for the energy prediction competition. *ASHRAE Transactions*, 100(2):1053–1062.

Mackay, D. J. C. (1997). Gaussian Processes: A replacement for supervised Neural Networks? Technical report, Cavendish Laboratory, Cambridge University, Cambridge, United Kingdom. Lecture notes for a tutorial at NIPS 1997.

MacKay, D. J. C. (1999). Comparison of Approximate Methods for Handling Hyperparameters. *Neural Computation*, 11:1035–1068.

Mackey, M. and Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289.

Neal, R. M. (1993). Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, University of Toronto, Toronto, Ontario.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, Heidelberg, Germany.

O'Hagan, A. (1977). On outlier rejection phenomena in bayes inference. *Journal of the Royal Statistical Society B*, 41:358–367.

O'Hagan, A. (1994). *Bayesian Inference*, volume 2B of *Kendall's Advanced Theory of Statistics*. Arnold, London, United Kingdom.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of IEEE*, 78:1481–1497.

Press, W., Flannery, B., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in C*. Cambridge University Press, Cambridge, United Kingdom, second edition.

Quiñonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. E. (2003a). Propagation of uncertainty in bayesian kernels models - application to multiple-step ahead forecasting. In *International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 701–704, Piscataway, New Jersey. IEEE.

Quiñonero-Candela, J., Girard, A., and Rasmussen, C. E. (2003b). Prediction at an uncertain input for gaussian processes and relevance vector machines - application to multiple-step ahead time-series forecasting. Technical Report IMM-2003-18, Technical University of Denmark, Kongens Lyngby, Denmark.

Quiñonero-Candela, J. and Hansen, L. K. (2002). Time series prediction based on the relevance vector machine with adaptive kernels. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 985–988, Piscataway, New Jersey. IEEE.

Quiñonero-Candela, J. and Winther, O. (2003). Incremental gaussian processes. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1001–1008, Cambridge, Massachussetts. MIT Press.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario.

Rasmussen, C. E. (2002). Reduced rank gaussian process learning. Unpublished Manuscript.

Rasmussen, C. E. and Kuss, M. (2004). Gaussian processes in reinforcement learning. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, Cambridge, Massachussetts. MIT Press.

Rubin, D. E. (1987). *Multiple Imputation for Nonresponse in Surveys.* John Wiley and Sons, Hoboken, New Jersey.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels.* MIT Press, Cambridge, Massachussetts.

Schwaighofer, A. and Tresp, V. (2003). Transductive and inductive methods for approximate gaussian process regression. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 953–960, Cambridge, Massachussetts. MIT Press.

Seeger, M. (2003). *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations.* PhD thesis, University of Edinburgh, Edinburgh, Scotland.

Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. In Bishop, C. M. and Frey, B. J., editors, *Ninth International Workshop on Artificial Intelligence and Statistics.* Society for Artificial Intelligence and Statistics.

Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy Gaussian process regression. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 619–625, Cambridge, Massachussetts. MIT Press.

Smola, A. J. and Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. In Langley, P., editor, *International Conference on Machine Learning 17*, pages 911–918, San Francisco, California. Morgan Kaufmann Publishers.

Svarer, C., Hansen, L. K., Larsen, J., and Rasmussen, C. E. (1993). Designer networks for time series processing. In Kamm, C., Kuhn, G., Yoon, B., Chellappa, R., and Kung, S., editors, *IEEE Workshop on Neural Networks for Signal Processing*, pages 78–87, Piscataway, New Jersey. IEEE.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society B*, 58(1):267–288.

Tikhonov, A. N. and Arsenin, V. Y. (1977). *Solutions of Ill-posed Problems*. W. H. Winston, Washington, D.C.

Tipping, M. E. (2000). The relevance vector machine. In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems 12*, pages 652–658, Cambridge, Massachussetts. MIT Press.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244.

Tipping, M. E. and Faul, A. C. (2003). Fast marginal likelihood maximisation for sparse bayesian models. In Bishop, C. M. and Frey, B. J., editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Tresp, V. (2000). A bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

Tresp, V., Ahmad, S., and Neuneier, R. (1994). Training neural networks with deficient data. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 128–135, San Francisco, California. Morgan Kaufmann Publishers.

Wahba, G., Lin, X., Gao, F., Xiang, D., Klein, R., and Klein, B. (1999). The bias-variance tradeoff and the randomized GACV. In Kerns, M. S., Solla, S. A., and Cohn, D. A., editors, *Advances in Neural Information Processing Systems 11*, pages 620–626, Cambridge, Massachussetts. MIT Press.

Weigend, A. S., Zimmermann, H. G., and Neuneier, R. (1996). Clearning. In Refenes, A.-P. and Abu-Mostafa, Y., editors, *Neural Networks in Financial Engineering*, pages 511–522, Singapore. World Scientific Publishing Company.

Williams, C. (1997a). Computation with infinite neural networks. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems 9*, pages 295–301, Cambridge, Massachussetts. MIT Press.

Williams, C. (1997b). Prediction with gaussian processes: From linear regression to linear prediction and beyond. Technical Report NCRG/97/012, Dept of Computer Science and Applied Mathematics, Aston University, Birmingham, United Kingdom.

Williams, C. and Rasmussen, C. E. (1996). Gaussian proceses for regression. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 514–520, Cambridge, Massachussetts. MIT Press.

Williams, C., Rasmussen, C. E., Schwaighofer, A., and Tresp, V. (2002). Observations of the nyström method for gaussiam process prediction. Technical report, University of Edinburgh, Edinburgh, Scotland.

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, Massachussetts. MIT Press.

Wipf, D., Palmer, J., and Rao, B. (2004). Perspectives on sparse bayesian learning. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, Cambridge, Massachussetts. MIT Press.