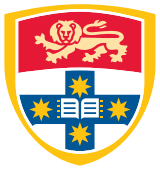
THE UNIVERSITY OF
SYDNEY

Function Embeddings for Multi-modal Bayesian Inference

Lachlan McCalman

A thesis submitted in fulfilment
of the requirements of the degree of
Doctor of Philosophy



THE UNIVERSITY OF
SYDNEY

Australian Centre for Field Robotics
School of Information Technology
University of Sydney

October 2014

Abstract

Lachlan McCalman
The University of Sydney

Doctor of Philosophy
October 2014

Function Embeddings for Multi-modal Bayesian Inference

Tractable Bayesian inference is a fundamental challenge in robotics and machine learning. Standard approaches such as Gaussian process regression and Kalman filtering make strong Gaussianity assumptions about the underlying distributions. Such assumptions, however, can quickly break down when dealing with complex systems such as the dynamics of a robot or multi-variate spatial models.

In this thesis we aim to solve Bayesian regression and filtering problems without making assumptions about the underlying distributions. We develop techniques to produce rich posterior representations for complex, multi-modal phenomena. Our work extends kernel Bayes' rule (KBR), which uses empirical estimates of distributions derived from a set of training samples and embeds them into a high-dimensional reproducing kernel Hilbert space (RKHS). Bayes' rule itself occurs on elements of this space.

Our first contribution is the development of an efficient method for estimating posterior density functions from kernel Bayes' rule, applied to both filtering and regression. By embedding fixed-mean mixtures of component distributions, we can efficiently find an approximate pre-image by optimising the mixture weights using a convex quadratic program. The result is a complex, multi-modal posterior representation. We demonstrate the utility of our algorithm on several motion estimation problems in robotics.

Our next contributions are methods for estimating cumulative distributions and quantile estimates from the posterior embedding of kernel Bayes' rule. Such information is critical for quantifying risk, and hence for using the output of inference in a decision-making process. We examine a number of novel methods, including those based on our density estimation techniques, as well as directly estimating the cumulative through use of the reproducing property of RKHSs. The performance of these techniques is compared against well-known conditional quantile estimation algorithms on standard machine learning datasets.

Finally, we develop a novel method for scaling kernel Bayes' rule inference to large datasets, using a reduced-set construction optimised using the posterior likelihood. This method retains the ability to perform multi-output inference, as well as our earlier contributions to represent explicitly non-Gaussian posteriors and quantile estimates. We apply our algorithm to a number of standard datasets used in large scale kernel-based regression, and also demonstrate competitive performance with comparable techniques from the Gaussian process literature.

Declaration

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the University or other institute of higher learning, except where due acknowledgement has been made in the text.

Lachlan McCalman

1st of October, 2014

To Heather and Iain.

Contents

1. Introduction 1
   - 1.1. Motivation . . . 3
   - 1.2. Problem Statement . . . 8
   - 1.3. Contributions . . . 9
   - 1.4. Thesis Structure . . . 10

2. Background 12
   - 2.1. Bayes' Theorem . . . 12
     - 2.1.1. Example . . . 13
     - 2.1.2. Numerical Considerations of Bayes' Theorem . . . 14
     - 2.1.3. Kernel Bayes' Rule . . . 14
   - 2.2. Vector Spaces . . . 14
     - 2.2.1. Bases . . . 16
   - 2.3. Function Spaces . . . 17
     - 2.3.1. Banach Spaces . . . 17
     - 2.3.2. Hilbert Spaces . . . 18
   - 2.4. Kernels . . . 20
     - 2.4.1. Reproducing Kernel Hilbert Spaces . . . 21
     - 2.4.2. RKHS Inner Product and Basis . . . 21
     - 2.4.3. Finite Subspaces of H . . . 22
     - 2.4.4. RKHS Distance Measure . . . 23
   - 2.5. Example Positive-Definite Kernels . . . 23

2.6. RKHS of Stochastic Functions . . . 27
2.6.1. Characteristic Kernels . . . 29
2.6.2. Empirical Approximation to the Mean Map . . . 30
2.7. RKHS Operators . . . 31
2.7.1. Cross-Covariance Operators . . . 33
2.7.2. Conditional Probabilities . . . 34
2.8. Kernel Probability Laws . . . 36
2.8.1. Kernel Sum Rule . . . 36
2.8.2. Kernel Product Rule . . . 37
2.9. Kernel Bayes' Rule . . . 38
2.9.1. Empirical Estimate . . . 39
2.9.2. Example . . . 42
2.10. Summary . . . 44

3. Multi-modal Regression and Filtering 46
3.1. Motivation . . . 46
3.2. Contributions . . . 47
3.3. Related Work . . . 48
3.4. Kernel Bayes' Rule Filtering . . . 52
3.4.1. Empirical Estimate . . . 53
3.5. Kernel Bayes' Rule Posterior Recovery . . . 54
3.5.1. The Pre-image Problem . . . 55
3.5.2. Fixed Kernel Mixture Pre-image Method . . . 57
3.5.3. Mixture of Kernels Posterior Estimates . . . 60
3.5.4. Normed-Weights Posterior Estimation . . . 62
3.5.5. The Multi-Modal Kernel Bayes' Rule Algorithm . . . 64
3.5.6. Computational Complexity . . . 65
3.6. Parameter Learning . . . 67
3.6.1. Implementation . . . 69
3.7. Experiments . . . 70
3.7.1. Comparison Algorithms . . . 71
3.7.2. Experiment 1: Multi-modal Tracking Simulation . . . 74

3.7.3. Experiment 2: Inertial Slot Car 80
3.7.4. Experiment 3: Pedestrian Velocity Field 83
3.8. Summary 87

4. Conditional Quantile Estimation from KBR Posterior Embeddings 89
4.1. Motivation 89
4.2. Contributions 90
4.3. Related Work 92
4.4. Background 94
4.4.1. Probability Measures 94
4.4.2. Quantiles 95
4.4.3. Quantile Regression 96
4.5. Measure Estimates from RKHS Embeddings 99
4.5.1. Optimal Approximate Embeddings 101
4.5.2. Optimal Empirical Measure Estimate 101
4.6. Smooth Measure Approximations 103
4.6.1. Example: Cumulative distribution 104
4.7. CDF-From-PDF Estimation Techniques 105
4.8. Overview of CDF Estimation Algorithms 107
4.9. KBR Quantile Estimation 108
4.9.1. Parameter Learning 108
4.9.2. Summary of Algorithms 109
4.10. Experiments 111
4.10.1. Our Algorithms 112
4.10.2. Comparisons 113
4.10.3. Results 113
4.11. Summary 118

5. Scalable Multi-Modal Regression 120
5.1. Motivation 120
5.2. Contributions 121
5.3. Related Work 122

5.4. Sparse Multi-modal Kernel Bayes' Rule 125
5.4.1. Computational Complexity 126
5.4.2. Example 127
5.4.3. Parameter Learning 127
5.5. Other MKBR Scaling Methods 130
5.5.1. Low-Rank Gram Matrix Approximation 131
5.6. Experiments 134
5.6.1. Datasets 134
5.6.2. Comparisons 137
5.6.3. Experimental Setup 138
5.6.4. Results 139
5.7. Summary 142

6. Conclusions 144
6.1. Summary of Contributions 145
6.2. Future Work 148

A. Foundations of Probability Theory 151
A.0.1. Cox's Axioms 152
A.0.2. Cox's Theorem 154
A.0.3. Laws of Probability 155
A.0.4. Continuous Sets 156

B. Full Experimental Results 157

Acknowledgments 161

List of Symbols

$\langle \cdot, \cdot \rangle$	The inner product of two vectors
$\| \cdot \|$	The norm of a vector
$\{\mathbf{e}_i\}$	A set of basis vectors of a vector space
A	A matrix in the MKBR pre-image cost function
$\boldsymbol{\alpha}$	A vector in $\mathbb{R}^N$
B	A matrix in the MKBR pre-image cost function
$\boldsymbol{\beta}$	A vector in $\mathbb{R}^N$
$\mathbb{C}$	The set of complex numbers
$C(a)$	The cumulative distribution function $P(x \leq a)$
$C_{X\|Y}$	Conditional embedding operator of X given Y
$\hat{C}_{X\|Y}$	Empirical estimate of the conditional embedding operator
C_{XY}	Cross-covariance operator between X and Y
$\hat{C}_{XY}$	Empirical estimate of the cross-covariance operator
c	Normalising constant to ensure kernel integral is unity
$\chi_\Omega(\cdot)$	The indicator function on a set Ω
$\hat{\chi}_\Omega(\cdot)$	Smoothed kernel approximation to the indicator function
D	A Boolean observation from a feature detector
$D(\mathbf{v})$	The diagonal matrix associated with a vector $\mathbf{v}$
$E[\cdot]$	The expectation operator
δ	Regularisation parameter for the inversion of G_{YY}
$\delta(\cdot)$	The Dirac delta function
$\delta_x[\cdot]$	The RKHS evaluation functional for the variable x
ε	Regularisation parameter for the inversion of G_{XX}

$\mathcal{F}$	An arbitrary field
f	A function element of an RKHS
G	A Gram matrix
G_{XY}	A Gram matrix over the random variables X and Y
g	A function element of an RKHS
$\boldsymbol{\gamma}$	A vector in $\mathbb{R}^N$
H	A Hilbert space or reproducing kernel Hilbert space (RKHS)
H	A discrete hypothesis
K_ν	The modified Bessel function
$k(\cdot,\cdot)$	A positive-definite kernel function
$\mathbf{k}_X^T[\cdot]$	The function $(k(x_1,\cdot),\ldots,k(x_N,\cdot))$ for a set $\{x_i\}_{i=1}^N$ and kernel k
L	Lower Cholesky factor of a matrix
$L_2[a,b]$	The space of twice-differentiable function on the interval (a,b)
L_τ	The pinball loss for the τ-th quantile
$\mathcal{A}$	The Fourier transform of a kernel function
λ	The regularisation term of the MKBR pre-image cost function
ℓ^2	The space of square-summable sequences
$M(\Omega)$	The probability measure $M(\Omega) = P(x \in \Omega)$
$M_k[P(X);t]$	The k-th moment of $P(X)$ centred at t
$\mu[\cdot]$	The mean map from a set to an RKHS
$\mu^{-1}[\cdot]$	The inverse mean map
$\mathcal{N}(\cdot;\mu,\Sigma)$	A normal distribution with mean μ and covariance Σ
$O(\cdot)$	The order of computational complexity of an algorithm
Ω	A subset of the domain of a random variable $X \in \mathcal{X}$
$\boldsymbol{\omega}[X]$	RKHS measure embedding vector
$\mathcal{P}_X$	The set of all probability distributions on X
$\mathbb{P}(\cdot)$	The power set operator
$P(X)$	The probability density function of X
$\hat{P}(X)$	A distribution approximating $P(X)$
$P(X\|Y)$	The conditional probability density function of X given Y
$P(X\|Y=y)$	The conditional PDF evaluated at the observation y of Y

$P(y)$	The PDF $P(Y)$ evaluated at y
$\Phi(\cdot)$	A mapping from a set to a Hilbert space
$Q(X)$	A probability density function of X
$q(\tau)$	The τ-th quantile function
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^N$	The space of real tuples of length N
$\hat{R}_{X\|Y}$	The empirical KBR posterior operator
$\mathcal{R}$	An arbitrary set
$\mathcal{S}$	An arbitrary set
Σ	The covariance matrix of a Normal distribution
σ	The kernel width parameter, or the standard deviation of a Gaussian
t	An index variable denoting discrete time steps
τ	The fraction of probability mass enclosed by a quantile function
ϑ	Unknown parameters determined by a training algorithm
U	A random variable
u	An observation of U
υ	The Matérn kernel smoothness parameter
$\mathcal{U}$	An arbitrary set
$\mathbf{w}$	A vector in $\mathbb{R}^N$
$\mathbf{x}$	A vector in $\mathbb{R}^N$
X	A random variable
x	An observation of X
$\mathbf{y}$	A vector in $\mathbb{R}^N$
Y	A random variable
y	An observation of Y

Chapter 1

Introduction

Among the key attributes of intelligence is the ability to create a model of the surrounding environment from unreliable, even conflicting observations that is powerful enough to generalise and make predictions.

Such models inform the actions we take at every scale; from retrieving a coin from behind the couch to enacting global emissions targets to curtail climate change. The brains of humans and animals have been solving this problem to varying degrees for millions of years in order to survive and procreate. It is only in the last few hundred years that mathematics has caught up, to the extent that we now have a formal system with codified rules for this kind of reasoning under uncertainty. That formalism is the theory of probability. It develops the heuristics of human learning into a uniquely valid system for inductive inference. Probability theory extends Aristotelian logic, which is the much older system of reasoning when we only deal with absolutes of truth and falsity, into the realm of the uncertain— the realm of the real world. As such, developing probability theory is building the tools to help humans make the right decisions. It is a supremely important task.

Bayes' theorem is the equation that describes inductive inference with probabilities. To use it to perform probabilistic inference requires us to explicitly enumerate many of our underlying assumptions. We must list all relevant states of the world we consider possible, assign each one a prior probability of being correct, and collate them into a set of mutually exclusive hypotheses. When we consider a new piece of evidence, we must define a likelihood function that computes the chance that evidence would be observed under each one

of our hypotheses. From these results we are then able to compute our posterior belief in the validity of each of our hypotheses; that is, the extent to which we believe each of our hypotheses is the true state of the world.

As might be expected, the two human endeavours that have embraced Bayesian reasoning particularly strongly are those attempting to quantify and reproduce intelligence itself; robotics and machine learning. Mobile robots receive continuous streams of data from multiple sensors which must be fused into a coherent model of the world. Visual data from a camera, encodings from motors in the joints or wheels, inertial information from accelerometers and gyroscopes, laser range-finding and radar, may all provide valuable information about the outside world, but each comes with its own uncertainties. Combining these measurements with Bayes' rule to jointly estimate a map of the surroundings and the robot's position in that map has been extremely successful: the ubiquitous Simultaneous Localization and Mapping (SLAM) [27, 8].

Machine learning also now routinely uses Bayesian methods to solve a wide variety of real world problems. General problems such as regression, classification and clustering have been tractably solved in both continuous and discrete cases, with varying amounts of prior knowledge [12]. Computer vision problems such as segmentation or classification of images, or registration of multiple images together [116] have also received the Bayesian treatment, along with document processing such as topic modelling or spam filtering [99]. The explosion of popularity of the internet has driven recent progress in Bayesian techniques for problems such as preference elicitation [18], and relationships in social networks.

The overarching aim of the work in this thesis is to advance Bayesian non-parametric inference in continuous spaces, particularly focusing on non-Gaussian, multi-modal probability distributions that often appear in robotics and machine learning problems. Our work builds on a method for Bayesian inference called kernel Bayes' rule (KBR), a very general non-parametric approximation of Bayes' rule that attempts to relax the restrictions on a particular prior and likelihood distribution. The method works by representing distributions as points in a high-dimensional space of functions, and then performing inference entirely inside this space. The space itself is defined implicitly through the choice of a particular kernel function, sharing many characteristics with the covariance function in Gaussian processes. Tractable approximate inference is possible inside this space of func-

tions. We extend kernel Bayes' rule in a number of ways: adding the ability to estimate multi-modal posteriors in the form of mixture distributions, constructing algorithms for estimating cumulative and quantile distributions, and scaling the techniques to very large datasets.

1.1. Motivation

If Bayes' rule really is a universally applicable tool for performing quantified logical reasoning under uncertainty, why does humanity not exist in an enlightened utopia where reason and evidence trump bias and superstition? There are many answers to this question— and most are suited more to a thesis in psychology than one in machine learning. That being said, the most fundamental problem is that for many real-world problems, the computational cost in computing posteriors using Bayes' rule is impossibly large. In particular, the requirement to sum likelihood contributions from every possible model under consideration requires computing high-dimensional integrals in the continuous case, or explicitly enumerating vast combinatoric spaces in the discrete case. In the continuous case, exact integrals exist for only a small number of prior and likelihood distributions and approximate integration techniques rapidly break down in high-dimensional settings. In the discrete case, adding variables to the hypothesis space causes a combinatoric increase in the number of possible states, and soon outgrows the abilities of modern computer hardware.

By way of example, explicitly computing probabilities over a person's possible ranking of 50 films requires considering roughly the same number of hypotheses as there are atoms in the Milky way. To say this is beyond our current computational capabilities is an understatement— a maximally efficient computer operating at the Landauer limit [72] would expend at least 10^{45} Joules to perform this computation, around the energy released in a supernova [61] and close to the total mass-energy of planet earth.

However, these tractability problems are secondary to a more fundamental issue; that of writing down an appropriate prior or likelihood function in the first place. In general, determining these functions systematically can itself be a difficult and probably intractable inference problem.

One approach that has seen successful application is to learn the prior and likelihood dis-

tributions themselves from data, and then restrict the class of possible posteriors to make inference tractable. This type of inference is called non-parametric, as the complexity of the model can grow with additional input, rather than being fixed from a set of parameters defined *a-priori*.

In the continuous domain (which is the focus of this thesis), a canonical example of a non-parametric inference system is the Gaussian process (GP). This model considers the space of hypotheses of functions, with properties determined by a covariance function that controls the spatial dependence of the function values. GPs assume that prior and likelihood distributions are Gaussian, and then approximate these with a set of training samples. The Gaussianity assumption allows for closed-form integration over the hypotheses space, yielding a tractable inference method that has been applied to many inference problems in machine learning, robotics, geophysics and ecology [98].

Given no other information, making a Gaussianity assumption is a justifiable choice; it is a maximum entropy distribution for a given mean and variance, and more importantly, the central limit theorem ensures that many distributions encountered in nature will be approximately Gaussian [52]. However, there are still plenty of examples of problems where the prior and likelihood distributions are more complex. Any kind of multi-modal behaviour, for instance, already requires a relaxing of this assumption. In robotics, we see this particularly with distributions derived from range-only or bearing-only sensors, or for motion models with unknown characteristics like the friction under-foot. In machine learning, there is multi-modal behaviour in in problems such as matching, preference elicitation, and inverse-kinematics [12]. Inference in geostatistics is often multi-modal [60], owing to the large null-space associated with sensor modalities such as gravity and electromagnetics. For a given set of observations with these sensors, there exists an infinite number of different subterranean structures with a high likelihood.

A simple example of the difference between a uni-modal (Gaussian process) estimate and a multi-modal estimate using our techniques on the same data is illustrated in figure 1.1. In this case, a GP is forced to explain the data with a large variance, obscuring the detail that a multi-modal estimate can capture. It is important to note that the GP places a large degree of plausibility in regions that have no data.

This behaviour can have very real implications, especially when the multi-modal behaviour arises in the context of decision making. One classic example from robotics is

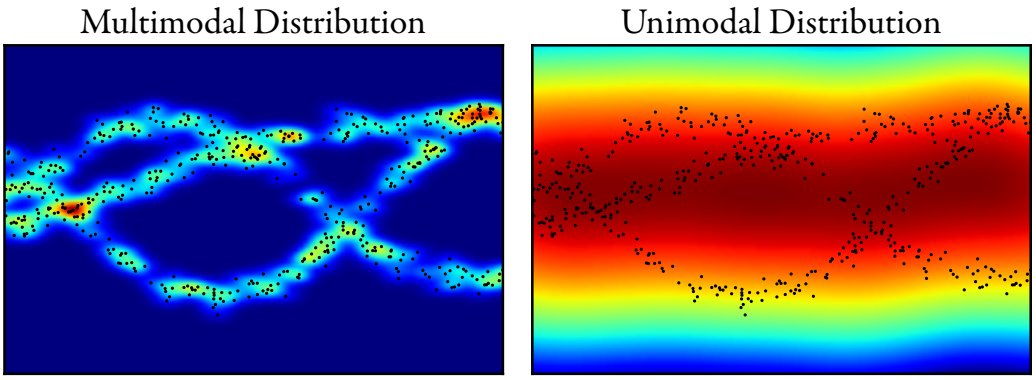


Figure 1.1.: Uni-modal distributions are forced to blur out details that a multi-modal distribution could represent. (Right) A Gaussian process estimate of a multi-modal distribution. (Left) A multi-modal estimate of the same data generated by the techniques described in this thesis.

tracking a robot as it moves around an obstacle under periodic observation. This example is illustrated in figure 1.2. Here a robot must drive around an obstacle in its path. However, we do not actually observe the robot during this manoeuvre, and so we must account for the possibility that it deviated to its left or its right. The resulting predictive distribution has two modes, corresponding to the two possible paths the robot can take. A Gaussian or uni-modal estimation of the predictive distribution is forced to average out these two explanations, and place a single mode of the prediction in the middle of the two paths. Unfortunately, this places the robot in a state we know to be quite unlikely; in the middle of the obstacle itself.

The principle of this simple example applies to much more general inference and regression problems. Wherever we have strong multi-modal behaviour, a uni-modal estimate of that behaviour will necessarily place large probability mass in unlikely regions.

A multi-dimensional state space poses another challenge to performing non-parametric inference. Traditional Gaussian processes estimate only one-dimensional functions. The so-called multi-task problem essentially requires stacking these one-dimensional estimators. The development of techniques to encode complex, non-linear relationships between the different output dimensions is ongoing [125]. It is also confounded by the fact that many complex systems have a high-dimensional state-space, and posterior distributions

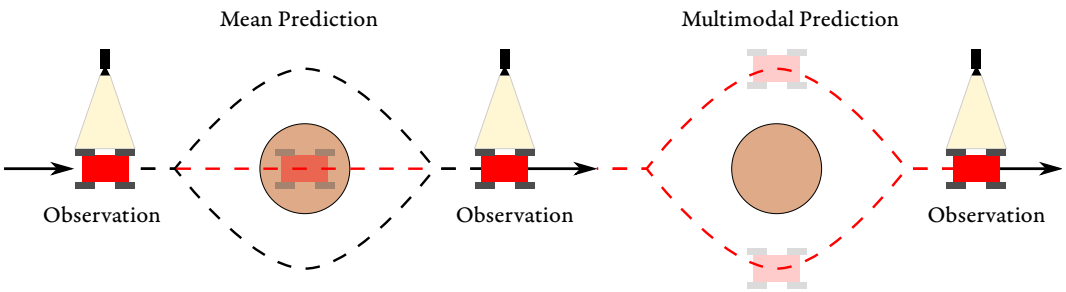


Figure 1.2.: Robot trajectory estimation from two observations. (Left) A uni-modal posterior will force the mean and the mode to coincide. In this example, the mean trajectory of the robot is actually through the obstacle. (Right) A multi-modal posterior prediction will account for the two possible states, correctly predicting two means for the robot on either side of the obstacle in this case.

over that state-space are a requisite for probabilistic decision making. Non-parametric methods are suited to this case, because we often have no analytic expression for the state evolution. At best, we might have a set of differential equations approximating the behaviour of the system. Figure 1.3 illustrates such a system— the Lorenz equations, which model several real-world phenomena including convection currents fluids. The state-space is three-dimensional, and the dimensions are linked by non-linear differential relationships. Additionally, the system is sensitively dependent on initial conditions, and observes unstable orbits of one of two points (visible as the centre of the two rings in the diagram). In that case, there is a high probability that the state will be orbiting one or other of these nodes, but a low probability that it will be found in the intervening space. This is exactly the multi-modal situation analogous to our obstacle-avoiding robot.

Assuming these issues have been addressed and a reasonable posterior distribution for a problem has been computed, the next task is often to make a decision or take an action based on this distribution. As such it may be important to quantify the risk associated with a particular outcome, rather than simply estimating its probability. Given a fixed bound on this risk, the quantile of a distribution gives a representative value of the unknown quantity that will not be exceeded within the bound. For instance, given a 95% confidence, the associated quantile is the value by which the unknown quantity is bounded above with 0.95 probability. In figure 1.4 we see how using quantile information allows for safe robot

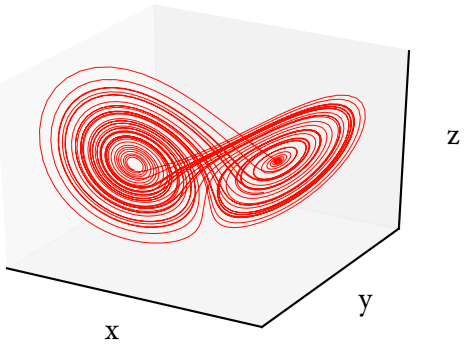


Figure 1.3.: A multi-dimensional state-space is difficult to model with standard non-parametric methods such as Gaussian processes, especially when the variables are non-linearly related as in this Lorenz state space example.

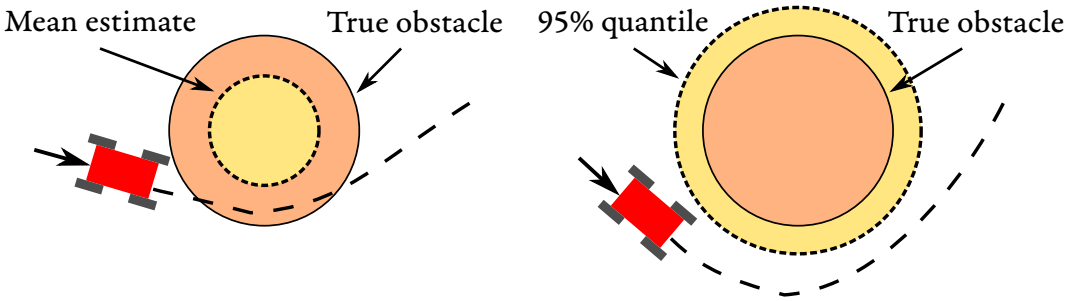


Figure 1.4.: Accurate estimation of quantiles is critical for safe planning.

motion planning. If we are unsure about the radius of an obstacle, by using a quantile estimate we can path plan to reduce the risk of collision to an acceptable level of our choice. Ideally, we would also like to be able to compute the confidence intervals and quantiles efficiently, to enable online decision making for robots. This may preclude first computing the posterior distribution and then numerically integrating.

Computational efficiency more generally is one of the significant barriers to overcome in implementing many non-parametric methods. Traditional algorithms such as the Gaussian processes scale poorly with increasing numbers of input points. Tractable inference for large problem sizes is critical in fields such as long-term autonomy, continental geophysics, or global-scale modelling. A naïve approach to implementing any kernel-based

non-parametric method requires at least N^2 storage and N^3 computational time. Solving problems in this way becomes difficult on a personal computer with more than about 10,000 points.

The consequence of these issues is that a non-parametric method for performing Bayesian inference that is tractable for large datasets, and can model multi-dimensional, multi-modal distributions arising from complex phenomena, and recover quantile information, has the potential to be a very powerful and very widely applicable tool. The goal of this thesis is to develop such a tool.

1.2. Problem Statement

This thesis addresses three problems associated with the non-parametric estimate for performing Bayesian inference, the kernel Bayes' rule. The first is the problem of computing a posterior density estimate from the function-space embedding that is the output of the algorithm. This requires solving the 'pre-image' problem, or to reverse the mapping of distributions into elements of the function space. A unique solution does not exist in the general case. Additionally, as there are no underlying assumptions about the nature of the posterior distribution, any parametrisation of a density estimate must be very flexible.

The second problem this thesis addresses is how to recover estimates of conditional cumulative distributions and quantiles from the KBR posterior embedding. These tools are important when employing risk-based decision making, often encountered in robotics and machine learning. The ability to enforce non-crossing constraints on the quantiles would be desirable, as would the ability to recover estimates without first having to compute posterior densities.

The third problem this thesis addresses is how to scale KBR estimation to problems with large training datasets, beyond $N > 10,000$ points. Unfortunately, a standard KBR implementation requires inversion of an N^2 matrix, precluding these larger datasets. Therefore, there is a requirement to either approximate this inversion directly, or to use a more efficient underlying representation that still makes use of all the data, but has lower storage and computation requirements.

In summary, this thesis addresses the following three problems:

1. How to extract an estimate of the posterior distribution from kernel Bayes' rule.

2. How to directly estimate quantiles of the posterior, without necessarily estimating the density first.

3. How to scale the KBR algorithm to larger problems, beyond 10,000 samples.

1.3. Contributions

To address these problems, this thesis presents the following contributions:

A method to recover the full posterior estimate from an application of kernel Bayes' rule. Our method solves the pre-image problem by embedding a mixture distribution into the function space, and optimising the parameters of that distribution to match the posterior embedding. We develop a new cost function and training scheme to enable learning the algorithm's parameters, and apply it to a number of difficult motion estimation problems in robotics. These problems demonstrate the ability for our new algorithm to handle non-linear models and multi-modal posteriors in multiple output dimensions.

Methods to estimate the cumulative distribution, quantile and confidence intervals for a KBR posterior. We present two different methods for estimating these quantities. The first estimates the cumulative at a particular point directly, by using the reproducing property of the embedded function space. The second estimates the cumulative by integration of a density estimate. The first method is more computationally efficient, whilst the second guarantees the non-crossing constraint for quantiles is upheld. We demonstrate competitive performance for both algorithms on a number of standard machine learning datasets.

A method to scale the KBR algorithm to large datasets. Our method is a reduced-set construction, in that we create a smaller subset of the training data with which to represent the function embeddings. This reduced set is optimised over all training data using the posterior likelihood. The entire dataset is also used to generate the prior embedding. We compare our approach to the low-rank approximations of the original KBR paper, and to several algorithms in the Gaussian process literature. Our algorithm shows competitive performance, and is much more flexible in terms of both the dimensionality and the form

of the posterior.

1.4. Thesis Structure

After this introduction, chapter two of this thesis presents the required background material for understanding the kernel embeddings and the kernel Bayes' rule algorithm, and places them in the wider context of kernel methods in machine learning. We provide a brief refresher on Bayes' theorem itself, before delving into the basics of functional analysis. We build up from vector spaces, to Banach spaces, to Hilbert spaces, and finally to reproducing kernel Hilbert spaces (RKHS), which form the mathematical basis for KBR. Embedding probability distributions in RKHSs follows, before we derive kernel Bayes' rule itself.

Chapter three describes our first major contribution; the derivation of a new method for recovering multi-modal density estimates from KBR embeddings, which we denote as multi-modal kernel Bayes' rule, or MKBR. After describing related work, we take the reader through the principles of our technique. This introduces the fundamental idea of this chapter and the next, which is to embed parametric functions into the Hilbert space associated with KBR inference, and using those embeddings to extract information about the posterior. Next, we describe the density estimation algorithm itself, including the training and implementation. Finally, we report on a number of experiments concerned with estimating motion models in robotics, using both the regression and filtering variants of KBR.

Chapter four extends the theory on function embeddings to extract quantile and confidence interval information from the posterior. We begin by discussing optimal embeddings of known parametric functions into the RKHS, which allows us to derive an expression for the cumulative of a posterior embedding in terms of dot products in the RKHS. We also discuss variations of this algorithm that enforce smoothness in the estimates. Next, we consider quantile recovery, both through the previously outlined method, but also through direct estimation of the density as in chapter three. We compare these new methods to a number of existing conditional quantile estimation algorithms in the machine learning literature.

Chapter five outlines our third main contribution, which is the novel algorithm for increasing computational scalability of MKBR and our quantile extensions in the previous chapter. After describing related work, particularly in the Gaussian process literature, we present our technique, first of the reduced-set construction, and then the associated optimisation. We first apply the algorithm to a toy problem to demonstrate its properties, then compare its performance against big-data GP algorithms on a number of large datasets.

Finally, chapter six concludes the thesis, considers the implications of the results, and discusses future work. An appendix provides additional information on proofs, software architecture and any useful mathematical results for working in the area.

Chapter 2

Background

The goal of this chapter is to introduce the reader to the mathematical background of kernel Bayes' rule and the extensions developed in this thesis. We begin with a brief re-cap of Bayes' rule itself, before making a foray into functional analysis to establish the important concepts of abstract vector spaces, and reproducing kernel Hilbert spaces in particular. From here, we examine embeddings of probability distributions in these spaces, and finally, the theorems required to construct kernel Bayes' rule.

2.1. Bayes' Theorem

Bayes' theorem is a direct consequence of the product and sum rules of probability. It describes the method for updating the prior probability $P(X)$ of a random variable X through making an observation y of another variable Y, assuming we know the likelihood $P(Y|X)$. From the product rule we have $P(Y|X)P(X) = P(X|Y)P(Y)$ and hence

$$P(X|Y=y) = \frac{P(Y|X)P(X)}{P(y)}. \tag{2.1}$$

For a discrete random variable X, the marginal distribution $P(y)$ is a sum over all possible values of x, $P(y) = \sum P(y|X=x)P(x)$. If X is continuous, then the sum becomes an integral; $P(y) = \int P(y|X=x)P(x)\,\mathrm{d}x$ and so

$$P(X|Y=y) = \frac{P(Y|X)P(X)}{\int P(Y|X=x)P(x)\,\mathrm{d}x}. \tag{2.2}$$

2.1.1. Example

A T-800 model robot is charged with a face recognition task: finding a subject John Connor, known to be somewhere in the United States. The T-800's software facial recognition system has been trained to be 99% accurate, in that it correctly recognises the subject's face 99% of the time, and correctly returns a negative result 99% of the time when faced with another human. Whilst moving through a busy street in New York, the face detector goes off. Should the robot act upon this result?

Fortunately for the bystanders, this T-800 model will use Bayes' theorem to compute the probability. Let H_j be the hypothesis that John Connor is on the street, and H_n be that he is not. The positive observation from the feature detector is denoted D. We are given the likelihood model of the T-800 sensor; $P(D|H_j) = 0.99$ and $(D|H_n) = 0.01$. The robot has no prior information that the subject is nearby, so it is effectively sampling randomly from the whole population of the United States before taking the observation. The population of the United States at the time (1996) is approximately 270 million, so a reasonable prior is $P(H_j) = 3.7 \times 10^{-9}$. Computing the posterior:

$$P(H_j|D) = \frac{P(D|H_j)P(H_j)}{P(D|H_j)P(H_j) + P(D|H_n)P(H_n)} \tag{2.3}$$

$$P(H_j|D) = \frac{0.99 \times 3.7 \times 10^{-9}}{0.99 \times 3.7 \times 10^{-9} + 0.01 \times (1 - 3.7 \times 10^{-9})} \tag{2.4}$$

$$P(H_j|D) \approx 4 \times 10^{-7}. \tag{2.5}$$

Here we see that despite the T-800 face detector being 99% accurate, the robot's poor prior knowledge means it is still overwhelmingly likely that D was a false positive. The robot must now take more data to increase the probability of a successful detection.

2.1.2. Numerical Considerations of Bayes' Theorem

Though direct application of Bayes' theorem is tractable for simple cases such as the example above, computing the posterior probabilities for a real-world experiment rapidly becomes all but impossible without making strong assumptions about the problem structure. This is due to the denominator of Bayes' theorem. In the discrete case, as the number of hypotheses grows we quickly run out of memory and time to store the computation, whilst in the continuous case, the integral only exists for a few special classes of distributions.

2.1.3. Kernel Bayes' Rule

How then do we solve equation 2.1 in general, without restricting ourselves to particular distributions? The solution offered by kernel Bayes' rule is to map prior and likelihood distributions $P(X)$ and $P(Y|X)$ to points in a high-dimensional vector space through some map $\mu[\cdot]$, and find analogue rules for the product rule and sum rule for distributions. Using these rules, Bayes' rule can be applied in the vector space to find an element in that space corresponding to a posterior distribution. In theory, we can use the inverse mapping μ^{-1} to ascertain what the result of Bayes' theorem would have been had we applied it directly. This concept is illustrated in figure 2.1. In practice however, finding the 'pre-image' of the posterior element is difficult, and forms the subject of chapter three.

The mapping we use is between distributions and points in an abstract space similar to a vector space called a reproducing kernel Hilbert space [11], and will be more precisely defined in due course. Distributions become points in this space under the mapping, and the sum and product rules have analogous operators. The following section gives an introduction to RKHS theory, leading in to the kernel rules of probability and kernel Bayes' rule, whilst a schematic overview of the KBR algorithm is given in figure 2.2.

2.2. Vector Spaces

The Euclidean vector spaces over $\mathbb{R}^N$ are ubiquitous in mathematics, partially because of the degree that our own experience affords us intuition in these spaces, especially in 2- and 3-D. These vector spaces are only an example of a much more general class of mathematical

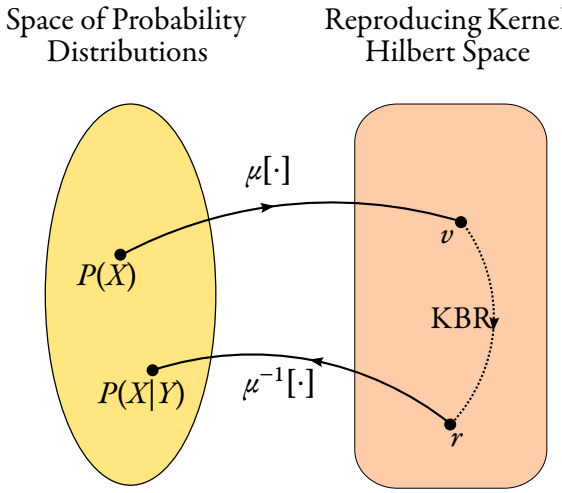


Figure 2.1.: An idealised version of kernel Bayes' rule algorithm. Note that in general, the inverse mapping $\mu^{-1}[\cdot]$ either does not exist or cannot be explicitly computed.

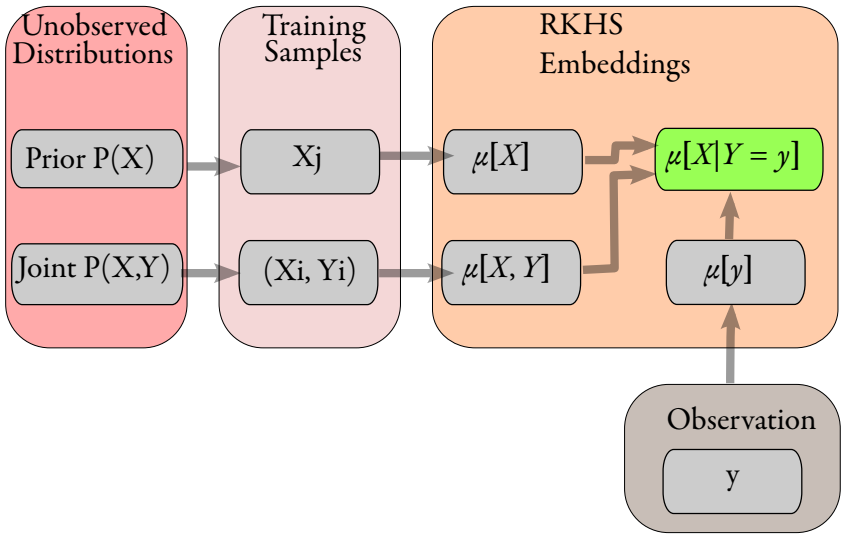


Figure 2.2.: Diagram of kernel Bayes' kule, including posterior density estimation. Note that the posterior embedding can be used as the prior to condition on a new observation, allowing continuous filtering inside the RKHS.

objects, which we can refer to as abstract vector spaces. Where as the Euclidean vector spaces are defined over power-sets of the real numbers, we are free to define a vector space over entirely different sets.

Abstract vector spaces are powerful, because, to an extent, the intuition we have developed in $\mathbb{R}^3$ carries over to these spaces as well. Formally, a vector space consists of a field $\mathcal{F}$, a set $\mathcal{V}$, and two binary operations, vector addition + and scalar multiplications. The elements of $\mathcal{F}$ are scalars, and the elements of $\mathcal{V}$ are vectors.

Definition 2.2.1 (Vector Space)**.** A vector space over a field $\mathcal{F}$ is a set $\mathcal{V}$ and two binary operations; vector addition (denoted by $\mathbf{u} + \mathbf{v}$ for $\mathbf{u}, \mathbf{v} \in \mathcal{V}$) and scalar multiplication (denoted by $a\mathbf{v}$ for $a \in \mathcal{F}, \mathbf{v} \in \mathcal{V}$), which satisfy the following axioms:

$$\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathcal{V}, \quad \mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}, \tag{2.6}$$

$$\forall \mathbf{u}, \mathbf{v} \in \mathcal{V}, \quad \mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}, \tag{2.7}$$

$$\forall \mathbf{v} \in \mathcal{V}, \exists 0 \in \mathcal{V} : \mathbf{v} + 0 = 0, \tag{2.8}$$

$$\forall \mathbf{v} \in \mathcal{V}, \exists -\mathbf{v} \in \mathcal{V} : \mathbf{v} + \forall \mathbf{v} \in \mathcal{V}, -\mathbf{v} = 0, \tag{2.9}$$

$$\forall \mathbf{u}, \mathbf{v} \in \mathcal{V}, a \in \mathcal{F}, \quad a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}, \tag{2.10}$$

$$\forall \mathbf{v} \in \mathcal{V}, a, b \in \mathcal{F}, \quad (a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}, \tag{2.11}$$

$$\forall \mathbf{v} \in \mathcal{V}, a, b \in \mathcal{F}, \quad a(b\mathbf{v}) = (ab)\mathbf{v}, \tag{2.12}$$

$$\forall \mathbf{v} \in \mathcal{V}, \quad 1\mathbf{v} = \mathbf{v}. \tag{2.13}$$

The Euclidean vector spaces set $\mathcal{V} = \mathbb{R}^N$, over either the real or complex numbers, $\mathcal{F} = \mathbb{R}$ or $\mathcal{F} = \mathbb{C}$.

2.2.1. Bases

A basis is a set of vectors used to build a vector space. Every element in the space can be constructed from a linear combination of the basis vectors, but no basis vector can be constructed from a linear combination of the others.

Definition 2.2.2 (Basis)**.** A basis $\{\mathbf{e}_i\} \in \mathcal{V}$ is a set of vectors indexed by elements of an

index set $i \in I$ such that

$$\mathbf{v} = \sum_{i \in I} a_i \mathbf{e_i}, \quad \forall \mathbf{v} \in \mathcal{V}, \tag{2.14}$$

$$\sum_{i \in I} a_i \mathbf{e_i} = 0 \longrightarrow a_i = 0, \quad \forall a_i \in \mathcal{F}. \tag{2.15}$$

In other words, a basis of $\mathcal{V}$ is a set of vectors that are linearly independent and span $\mathcal{V}$. Every vector space has a basis. The dimension of the vector space is equal to the cardinality of the index set I. For instance, the Euclidean vector space on $\mathbb{R}^3$ has an index of $\{1, 2, 3\}$ which is of cardinality 3. Larger spaces require larger index sets, until we reach an index set with infinite cardinality, such as the reals. The vector spaces we are interested in belong to this class. They are the function spaces.

2.3. Function Spaces

Function spaces are vector spaces in which the set of vectors $\mathcal{V}$ corresponds to a set of functions. The associated index set I is often infinite (for instance $\mathbb{R}^N$), making such vector spaces 'infinite-dimensional'.

The general class of infinite dimensional function spaces are not going to be particularly useful to us, because we have not yet developed any notion of an infinite sum in these spaces. So, for instance, even though we now have an infinite dimensional basis, we are only able to say that linear independence holds for any finite sum of the basis, and that every $\mathbf{v} \in \mathcal{V}$ can be constructed from a finite number of basis vectors.

In order to consider constructing a vector space from an infinite basis, we must explicitly ensure that every infinite sum of weighted basis vectors in $\mathcal{V}$ actually converges to an element in $\mathcal{V}$. Spaces with this property are called complete. It is to this subclass of vector spaces we now turn.

2.3.1. Banach Spaces

Banach spaces are complete vector spaces possessing a notion of length of a vector given by a norm operator. Banach spaces are complete in the sense that every absolutely converging

sequence of vectors in $\mathcal{V}$ converges to an element in $\mathcal{V}$. For infinite-dimensional function spaces, this property allows us to use infinite sums of basis vectors to construct elements of the space. From Banach spaces onwards, the field $\mathcal{F}$ associated with our vector space must be $\mathbb{R}$ or $\mathbb{C}$. For the sake of brevity, we will use the reals, noting only when a result is not directly analogous to $\mathbb{C}$ as well.

Definition 2.3.1 (Banach Space)**.** A Banach space is a vector space over $\mathbb{R}$ or $\mathbb{C}$ with a norm operator $\|\cdot\|$ satisfying

$$\|\mathbf{v}\| = 0 \iff \mathbf{v} = 0, \|\mathbf{v}\| > 0 \iff \mathbf{v} \neq 0, \tag{2.16}$$

$$\|a\mathbf{v}\| = |a|\|\mathbf{v}\| \forall a \in \mathcal{F}, \tag{2.17}$$

$$\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{u}\|, \tag{2.18}$$

and is complete, in that for every Cauchy sequence converges to an element of the space with respect to the norm.

The norm property in Banach spaces define vectors length, and has properties familiar from the well-known Euclidean vector spaces. They ensure that the zero vector is the only vector with zero length, and all other vectors have positive length. Multiplying a vector by a scalar multiplies the length of that vector by the modulus of that scalar, and finally, that the triangle inequality holds, which is that the shortest distance between two points is a straight line.

2.3.2. Hilbert Spaces

Hilbert spaces are a particular type of Banach space which possess a notion of angle. This is because Hilbert spaces are defined to include an inner product (or dot product in terms of the Euclidean vector spaces).

Definition 2.3.2 (Hilbert Space)**.** A Hilbert space is a Banach space $(\mathcal{F}, \mathcal{V})$ with an inner product operator $\langle\cdot,\cdot\rangle\colon \mathcal{V}\otimes\mathcal{V} \to \mathcal{F}$ satisfying the following properties for $\mathbf{u}, \mathbf{v} \in \mathcal{V}$ and

$a \in \mathcal{F}$:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \overline{\langle \mathbf{v}, \mathbf{u} \rangle}, \tag{2.19}$$

$$\langle a\mathbf{u}, \mathbf{v} \rangle = a\langle \mathbf{u}, \mathbf{v} \rangle, \tag{2.20}$$

$$\langle \mathbf{u}, \mathbf{u} \rangle > 0. \tag{2.21}$$

The norm of a Hilbert space is induced by the inner product

$$||\mathbf{u}|| = \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle}, \tag{2.22}$$

which also allows definition of distance between two points

$$||\mathbf{u} - \mathbf{v}|| = \sqrt{\langle (\mathbf{u} - \mathbf{v}), (\mathbf{u} - \mathbf{v}) \rangle} \tag{2.23}$$

$$= \sqrt{\langle \mathbf{u}, \mathbf{u} \rangle + \langle \mathbf{v}, \mathbf{v} \rangle - \langle \mathbf{u}, \mathbf{v} \rangle - \langle \mathbf{v}, \mathbf{u} \rangle}. \tag{2.24}$$

The angle between two vectors follows as

$$\cos\theta \equiv \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{||\mathbf{u}||||\mathbf{v}||}. \tag{2.25}$$

Hence, two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthogonal if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$:

$$\mathbf{u} \perp \mathbf{v} \Leftrightarrow \langle \mathbf{u}, \mathbf{v} \rangle = 0. \tag{2.26}$$

Hilbert spaces are an abstraction from the 3-D Euclidean vector space with which we commonly approximate the geometry of our universe. Some other examples include every finite-dimensional inner product space, the space of square-summable sequences l^2, and the space of twice-differentiable functions in a finite interval $L_2[a, b]$. From now on, we will be considering Hilbert spaces exclusively in our theory. Nonetheless, it is useful to understand exactly what intuitions do and do not apply to these abstract vector spaces, and how they relate to the familiar Euclidean 3-space.

2.4. Kernels

The mathematical tools in Hilbert space such as length and distance between points are useful when trying to generalise about a set $\mathcal{S}$ or understand its structure. If $\mathcal{S}$ is not itself a Hilbert space, we can still gain use of these tools by constructing a mapping from $\mathcal{S}$ into a different set $\mathcal{U}$ over which a Hilbert space can be defined. We would also then gain use of the intuition developed about $\mathbb{R}^3$.

Kernels are functions which allow us to construct just such a mapping between an arbitrary set $\mathcal{S}$ and a Hilbert space $H(\mathcal{U}, \langle , \rangle)$. There are two ways to achieve this end, the most common of which is the so-called 'kernel trick'[2, 14]. This method defines a kernel as a symmetric function $k(x, x'): \mathcal{S} \otimes \mathcal{S} \to \mathbb{R}$ between two elements an arbitrary set $\mathcal{S}$ and the reals. Suitable choices of k implicitly defines a mapping $\Phi: \mathcal{S} \to H$ from $\mathcal{S}$ to a Hilbert space H though the definition of the inner product:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle. \tag{2.27}$$

The condition on k to be a valid inner product is that k is *positive definite*:

Definition 2.4.1 (Positive-Definite Kernel). A positive-definite kernel on a set $\mathcal{S}$ is a symmetric function $\mathcal{S} \otimes \mathcal{S} \to \mathbb{R}$ which for all finite sets $\{x_i\}_{i=1}^N, x_i \in \mathcal{S}$, The matrix $G_{ij} = k(x_i, x_j)$ is positive-definite.

As long as analysis is restricted to operations built from inner products, $\mathcal{S}$ can be manipulated as if it were a Hilbert space, and without ever computing Φ explicitly. The kernel trick has enabled many types of algorithms is machine learning, including support vector machines (SVMs), kernel dimensionality reduction methods such as kernel principle component analysis and kernel independent component analysis, and a host of other 'kernel methods' [105].

Algorithms that cannot be couched in terms of an inner product require an explicit computation of the mapping Φ. For a given kernel, neither Φ nor the space H is uniquely determined [114]. However, there exists a particular choice of Φ which has several useful properties and forms the basis for our further work.

2.4.1. Reproducing Kernel Hilbert Spaces

One simple mapping $\Phi(x)$ can be constructed by Currying [10] a positive-definite kernel function. That is, by considering $k(x, x')$ not as a function of two variables, but of only one variable $x \in \mathcal{S}$, that returns a function of one variable x' through partial application,

$$k: \mathcal{S} \to [\mathcal{S} \to \mathbb{R}]. \tag{2.28}$$

For notational convenience, the function returned by k is often denoted $k(x, \cdot)$, and hence we can write $\Phi(x) = k(x, \cdot)$. Using this functional defines a *reproducing kernel Hilbert space* (RKHS) over $\mathcal{S}$.

Definition 2.4.2 (Reproducing Kernel Hilbert Space)**.** A reproducing kernel Hilbert space is a Hilbert space of functions from a set $\mathcal{S}$ to $\mathbb{R}$ in which for every $x \in \mathcal{S}$ and $f \in H$ there exists an evaluation functional δ_x which is bounded and continuous in x such that $\delta_x[f] = f(x)$.

A reproducing kernel $k(x, \cdot) \in H$ is then a function which for every $f \in H$,

$$\langle k(x, \cdot), f \rangle = f(x). \tag{2.29}$$

This equation is known as the reproducing property for an RKHS. Critically, if k is positive-definite, then the Hilbert space induced by $k(x, \cdot)$ will always be an RKHS, with the (unique) evaluation functional δ_x given by $k(x\cdot)$ itself.

The converse is also true— that is, every RKHS has a unique positive-definite kernel acting as the evaluation functional. This result is known as Moore-Aronszajn's theorem [11].

2.4.2. RKHS Inner Product and Basis

An RKHS defined through a set $\mathcal{S}$ and a reproducing kernel k is actually unique only up to isomorphism— that is, we still have not determined the feature space $\mathcal{U}$ over which H is defined. Nonetheless, if $\mathcal{S}$ is separable (i.e. contains a countable dense subset, I say), and Φ is continuous, then the set of all vectors $\Phi(x_i), i \in I$ forms a (countable) basis of $\mathcal{U}$ and hence H [114, 90]. Therefore, we can write any element $f \in H$ as a countable sum of these

basis vectors, weighted by a set of real coefficients $\{a_i\}_{i\in I}$:

$$\forall f \in H,\ \exists \{a_i\}_{i\in I}: \sum_{i\in I} a_i k(x_i, \cdot) = f. \tag{2.30}$$

Knowing this, for two arbitrary elements f and g of H we can write their inner product purely in terms of these basis vectors:

$$f = \sum_{i\in I} a_i k(x_i, \cdot), \tag{2.31}$$

$$g = \sum_{i\in I} b_i k(x_i, \cdot), \tag{2.32}$$

$$\langle f, g\rangle = \sum_{i,j\in I} a_i b_j k(x_i, x_j). \tag{2.33}$$

So, we have constructed an explicit mapping $\Phi = k(x, \cdot)$ between an arbitrary set $\mathcal{S}$ and an RKHS H. All the tools of these spaces are now available to analyse $\mathcal{S}$, as we can explicitly write elements of H in terms of basis functions. This is a very powerful concept, and will be used to construct the space inside which kernel Bayes' rule is defined.

2.4.3. Finite Subspaces of H

If the set $\mathcal{S}$ is finite of size N, then the index set of the basis is just $1 \ldots N$. If $\mathcal{S} = \{x_i\}_{i=1..N}$ then a basis set for H with kernel k is $\{k(x_i, \cdot)\}_{i=1\ldots N}$. In this case elements in H are finite linear combinations of basis vectors:

$$\forall v \in H,\ \exists \mathbf{a} \in \mathbb{R}^N: v = \sum_{i=1}^{N} a_i k(x_i, \cdot). \tag{2.34}$$

Just like in the Euclidean vector spaces, we can dispense with the basis vectors themselves and perform computations in terms of the weight vector $\mathbf{a}$. In fact, the Hilbert space is now isomorphic to $\mathbb{R}^N$ under the mapping $\mathbf{a} \leftrightarrow \sum_{i=1}^{N} a_i k(x_i, \cdot)$.

The inner product in H can be specified by the *Gram Matrix*, which is a matrix of inner

products of all pairs of basis vectors:

$$(G_{XX})_{ij} = k(x_i, x_j). \tag{2.35}$$

For two elements of H with weight vectors $\mathbf{u}$ and $\mathbf{v}$, the weight vector of their inner product is written in terms of G_{XX} as

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T G_{xx} \mathbf{v}. \tag{2.36}$$

2.4.4. RKHS Distance Measure

The RKHS norm enables evaluating distances between functions. However, it has an additional useful property, which is that it also constrains the point-wise difference between two functions. As the RKHS evaluation functional is bounded and continuous, for any two functions f and g in H there exists an M such that

$$\sup[|f(t) - g(t)|] \leq M \|f - g\|_H \quad \forall f, g \in H. \tag{2.37}$$

This demonstrates that the RKHS distance measure is a useful proxy for the similarity of two functions. For further information about RKHSs, an excellent short introduction can be found in [50] or [90]. For a longer treatment emphasising applications and particularly support vector machines try [105]. A more principled introduction with an emphasis on mathematical rigour is [11].

We now give some brief examples of positive definite kernels commonly used in machine learning.

2.5. Example Positive-Definite Kernels

Squared Exponential Kernel

One common positive-definite kernel is the squared exponential or Gaussian kernel [98]. This is a symmetric kernel, in the sense that $k(x, x') = k(x', x)$, and stationary, meaning $k(x, x')$ is a function only of the distance $\|x - x'\|$. The RKHS induced by this kernel

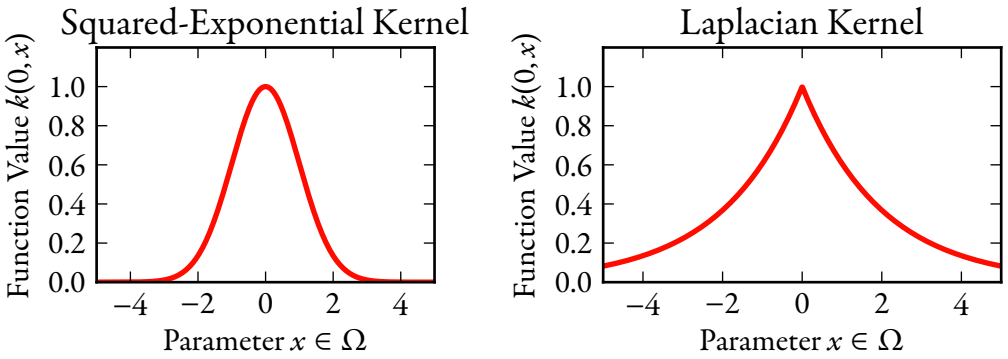


Figure 2.3.: Example kernels centred at $x = 0$. Left: The squared exponential kernel with $\sigma = 1$. Right: The Laplacian kernel with $b = 1$.

contains only infinitely differentiable functions. It is also sometimes referred to as the radial basis function (RBF) kernel, owing to its radial symmetry. The squared exponential kernel is plotted in 1-D in figure 2.3. Its analytic expression is an un-normalised multivariate Gaussian with a covariance matrix $2\sigma^2 I$:

$$k(x, x') = \exp\left[-\frac{\|x - x'\|^2}{2\sigma^2}\right]. \tag{2.38}$$

Matérn Kernels

Matérn kernels are a generalisation of the Gaussian kernel which are only finitely differentiable [98]. Like the Gaussian kernel, the Matérn family is stationary, able to be written as a function of $r = \|x - x'\|$;

$$k(r) = \frac{2^{1-\upsilon}}{\Gamma(\upsilon)}\left(\frac{\sqrt{2\upsilon}r}{l}\right)^{\upsilon} K_{\upsilon}\left(\frac{\sqrt{2\upsilon}r}{l}\right), \tag{2.39}$$

where K_{υ} is the modified Bessel function. Common special cases of this function are $\upsilon = 3/2$ and $\upsilon = 5/2$:

$$k_{\upsilon=3/2}(r) = \left(1 + \frac{\sqrt{3}r}{l}\right)e^{-\frac{\sqrt{3}r}{l}}, \tag{2.40}$$

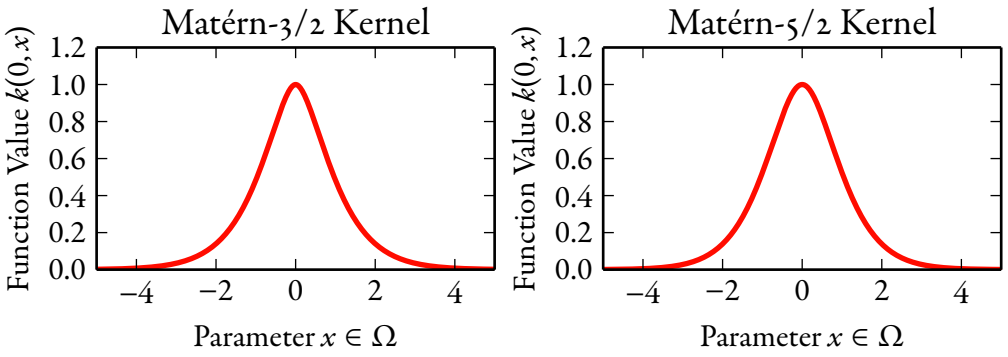


Figure 2.4.: Matérn kernels centred at $x = 0$. Left: $\upsilon = 3/2$, Right: $\upsilon = 5/2$.

$$k_{\upsilon=5/2}(r) = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) e^{-\frac{\sqrt{5}r}{l}}. \tag{2.41}$$

Setting $\upsilon = 1/2$ gives the exponential or Laplacian covariance function $k = \exp(r/l)$ plotted in figure 2.3. The Matérn kernel is plotted for some values of υ in figure 2.4. As $\upsilon \to \infty$, the Matérn kernels converge to the Gaussian kernel.

Piecewise-Polynomial Kernels

Polynomial kernels are a broad class of kernels defined through a polynomial on the input parameters. One useful series of these kernels are described in [98]. These kernels are stationary and approximately Gaussian, but are compact. This induces sparsity in the associated Gram matrix, and hence provides the potential for numerical savings in computations with large numbers of data points. In particular, the product of a sparse Gram matrix and a vector can be computed efficiently if the matrix is sparse.

The following are all positive-definite up to an input dimension D, and are $2q$ times

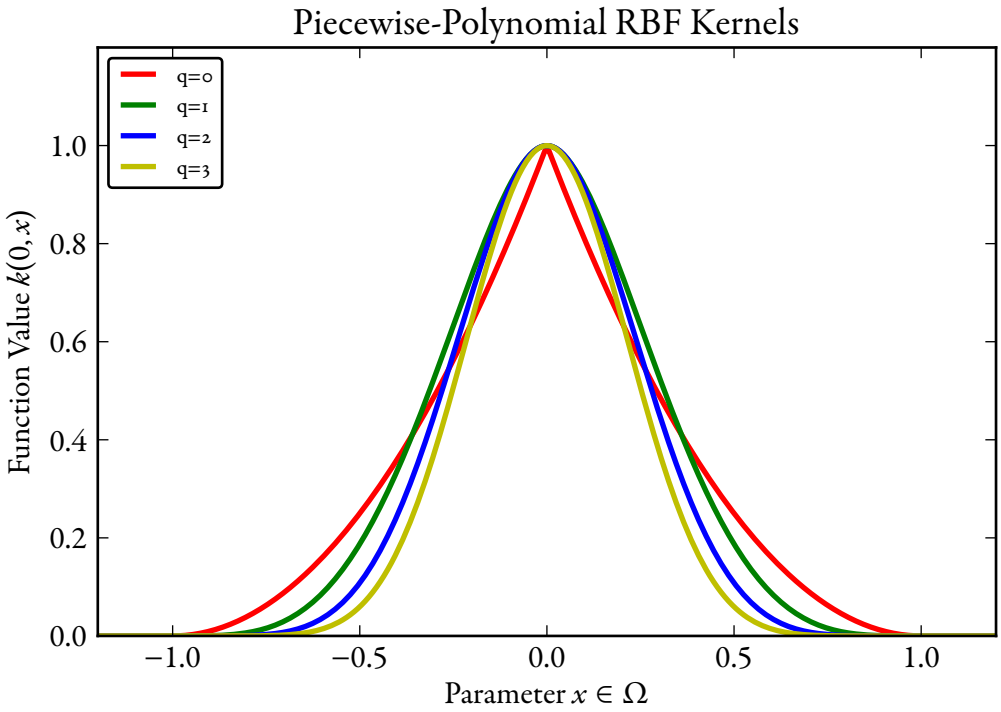


Figure 2.5.: Compact piecewise-polynomial RBF kernels.

differentiable. This series of kernels is plotted in figure 2.5.

$$k_{D,q=0} = (1-r)_+^j \tag{2.42}$$

$$k_{D,q=1} = (1-r)_+^{j+1}\left[(j+1)r+1\right] \tag{2.43}$$

$$k_{D,q=2} = (1-r)_+^{j+2}\left[(j^2+4j+3)r^2+(3j+6)r+3\right]/3 \tag{2.44}$$

$$k_{D,q=3} = (1-r)_+^{j+3}[(j^3+9j^2+23j+15)r^3 \tag{2.45}$$

$$+(6j^2+36j+45)r^2+(15j+45)r+15]/15 \tag{2.46}$$

where $j = \frac{D}{2} + q + 1$, and $(\cdot)_+ = \max(\cdot, 0)$. A selection of these kernels are plotted in figure 2.5.

Non-stationary Kernels

Non-stationary kernels cannot be written as a function of $||x - x'||$, hence the kernel function itself varies with location x or x'. This is an important class of kernels that allow for changes in the length scale or behaviour of the function in different subsets of the input space $\mathcal{S}$. However, these changes must be parametrised in the kernel, and learning those parameters can itself be a difficult inference problem. An example non-stationary kernel is illustrated in figure 2.6. A common non-stationary kernel is a generalisation of the Gaussian to allow explicit parametrisation of the kernel bandwidth σ as a function of the input space. This is known as the Paciorek kernel[89]:

$$k(x,y) = \prod_{d=1}^{D} \left(\frac{2 l_d(x) l_d(y)}{l_d^2(x) + l_d^2(y)} \right)^{\frac{1}{2}} \exp\left[- \sum_{d=1}^{D} \frac{(x_d - y_d)^2}{l_d^2(x) + l_d^2(y)} \right] \tag{2.47}$$

with $l_i(x)$ a length scale function which is arbitrary but must be strictly positive. An example of this kernel is illustrated in figure 2.6.

2.6. RKHS of Stochastic Functions

In the previous sections we outlined the properties of RKHSs defined over an arbitrary set $\mathcal{S}$. Now we will narrow our focus to consider the case when $\mathcal{S}$ is a set of probability distributions of a random variable X.

One map that can be used to link points in the space of distributions $\mathcal{P}_X$ over X with an RKHS $\mathcal{H}$ is the *mean map*, so called because elements of $\mathcal{H}$ are the expectation of the embedding $k(X, \cdot)$ of X. Intuitively, the mean map of X is the mean of all the possible RKHS elements corresponding to draws from X.

Definition 2.6.1 (Mean Map)**.** Given a random variable X with distribution $P(X)$ and an RKHS $\mathcal{H}$ with kernel k, the mean map $\mu[\cdot]: \mathcal{P}_X \to \mathcal{H}$ is the function

$$\mu[P(X)] = \mathrm{E}\left[k(X, \cdot)\right] \tag{2.48}$$

$$= \int k(x, \cdot) P(x) \, \mathrm{d}x. \tag{2.49}$$

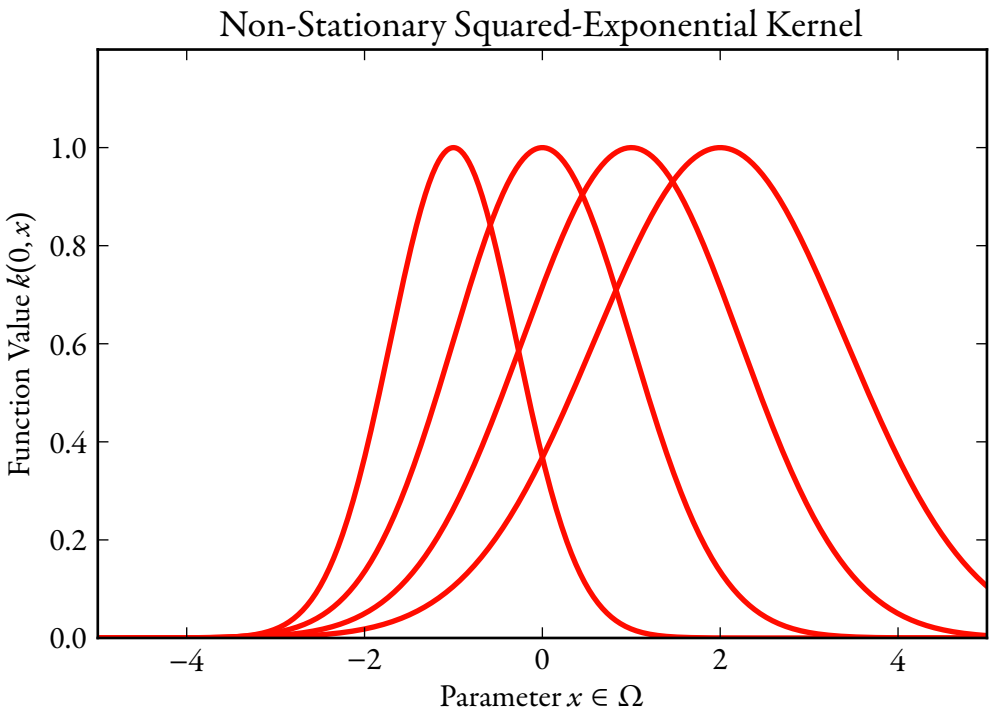


Figure 2.6.: Non-stationary RBF kernel with simple linearly decreasing length scale.

This definition implies the reproducing property, which allows us to recover expectations of functions in H over the random variable X. For $f \in H$,

$$\langle f, \mu[P(X)] \rangle = \mathrm{E}\left[f(x)\right]. \tag{2.50}$$

The mean map can preserve high-order moment information about $P(X)$ [31]. Consider the Gaussian kernel with unit bandwidth,

$$\begin{aligned}
\mu[P(X)](t) &= \int e^{-\|x-t\|^2} P(x)\,\mathrm{d}x, && (2.51)\\
&= \int \sum_{k=0}^{\infty} -\frac{(\|x-t\|^2)^k}{k!} P(x)\,\mathrm{d}x && (2.52)\\
&= \sum_{k=0}^{\infty} -\frac{E[\|x-t\|^2 k]}{k!} && (2.53)\\
&= \sum_{k=0}^{\infty} \frac{-1}{k!} M_{2k}[P(X); t], && (2.54)
\end{aligned}$$

where $M_{2k}[P(X); t]$ is the $2k$-th moment of $P(X)$ centred at t. Intuitively, we see that the mean map can preserve the complex structure of the distribution $P(X)$.

2.6.1. Characteristic Kernels

The mean mapping is a powerful tool to manipulate probability distributions or their empirical estimates inside an RKHS. However, using kernel embeddings as a proxy for distributions would require that different distributions not be mapped to the same element in the RKHS. In other words, the mean map $\mu[\cdot]$ should be one-to-one.

The subset of positive-definite kernels which induce this property on $\mu[\cdot]$ is known as the characteristic kernels [32].

Definition 2.6.2 (Characteristic Kernel)**.** Let $\mathcal{P}_X$ be the set of all probability distributions on the set S associated with a random variable X. A kernel k is characteristic over $\mathcal{P}_0 \subset \mathcal{P}_X$ if for all $P(X), Q(X) \in \mathcal{P}_0$,

$$\|\mu[P(X)] - \mu[Q(X)]\|_H = 0 \longrightarrow P(X) = Q(X). \tag{2.55}$$

A powerful result for determining which kernels are characteristic was developed in [113]. It is built from the characterisation of positive-definite kernels via Bochner's theorem [102];

Theorem 2.6.1 (Bochner's Theorem). *The kernel $k(\mathbf{r})$ is positive semi-definite, uniformly continuous and has $k(\mathbf{0}) = 1$, $\|k(\mathbf{r})\| \leq 1$ if and only if the spectral density $\Lambda(\omega) \geq 0$, where Λ is the Fourier transform of k;*

$$k(\mathbf{r}) = \int e^{i\mathbf{r}^T\omega}\Lambda(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}. \tag{2.56}$$

Sriperumbudur proved that positive-definite stationary kernels are characteristic if and only if their spectrum Λ has a support of all of $\mathbb{R}^k$:

Theorem 2.6.2. *Let $\mathcal{H}$ be an RKHS with kernel k. If k is a stationary and continuous positive-definite kernel on $\mathbb{R}^d$, then k is characteristic in for all probability distributions $P(X) \in \mathcal{P}$, if and only if $supp(\Lambda) = \mathbb{R}^d$, where Λ is the kernel spectrum [113].*

This result demonstrates that common kernels such as the Gaussian and Laplacian, as well as all compactly supported translation-invariant kernels on $\mathbb{R}^d$ such as the piecewise-polynomial kernel in section 2.5, are characteristic. Therefore, with these kernels the mean map $\mu[\cdot]$ is one-to-one, meaning distributions can safely be embedded in an RKHS with these kernels without information loss.

2.6.2. Empirical Approximation to the Mean Map

When the underlying distribution $P(X)$ is not available, it is still possible to create an approximate embedding using a set of finite samples $\{x_i\}_{i=1}^N \in \mathcal{S}$ drawn from $P(X)$. In this case the expectation becomes an empirical average over the samples:

$$\mu[X] \approx \mu[P(X)] \tag{2.57}$$

$$\mu[X] = \frac{1}{N}\sum_{i=1}^{N} k(x_i, \cdot) \tag{2.58}$$

$$= \frac{1}{N}\mathbf{k}_X^T[\cdot]\mathbf{1} \tag{2.59}$$

where $\mathbf{k}_X^T[\cdot] = (k(x_1, \cdot), ..., k(x_N, \cdot))$. For a given set of samples, this defines a finite-dimensional Hilbert space isomorphic to $\mathbb{R}^N$, as seen in section 2.4.3.

Given two arbitrary elements $f, g \in H$ with $H = (\{x\}_{i=1}^N, k)$, if $f = \mathbf{k}_X^T\mathbf{a}$ and $g = \mathbf{k}_X^T\mathbf{b}$, the inner product both in H and the associated Euclidean space over $\mathbb{R}^N$ is

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^T G_{XX} \mathbf{b} \tag{2.60}$$

where G_{XX} is the Gram matrix.

Moreover, if the kernel k is characteristic, then an N-sample empirical embedding converges to the true embedding (with high probability) at a rate $O(N^{-0.5})$ [112]. This allows us to use finite samples as proxies for unknown distributions, with the reassurance we can guarantee the embeddings converge to the true embedding as the number of sample points increases.

Figure 2.7 illustrates an example of an empirical approximation of a distribution embedded in an RKHS. It shows the empirical mean embedding for a three component Gaussian mixture in one dimension, created using 180 draws.

Apart from the work on conditional embeddings described in following sections, empirical embeddings of probability distributions in RKHS have been used for several applications in machine learning and statistics; in [108], these maps were used to test independence of two distributions (see also [42]), to perform covariate shift correction, and for feature extraction, and density estimation. Work has also been done on the two sample problem [41], and kernelised sorting [95].

2.7. RKHS Operators

The mean map embeds probability distributions in an RKHS, but to manipulate distributions within the space a set of RKHS operators is required. These operators are used to embed conditional distributions, and also to map between different RKHSs. Most importantly, they form the foundations of the kernel analogues for the rules of probability (reviewed in appendix A.0.3).

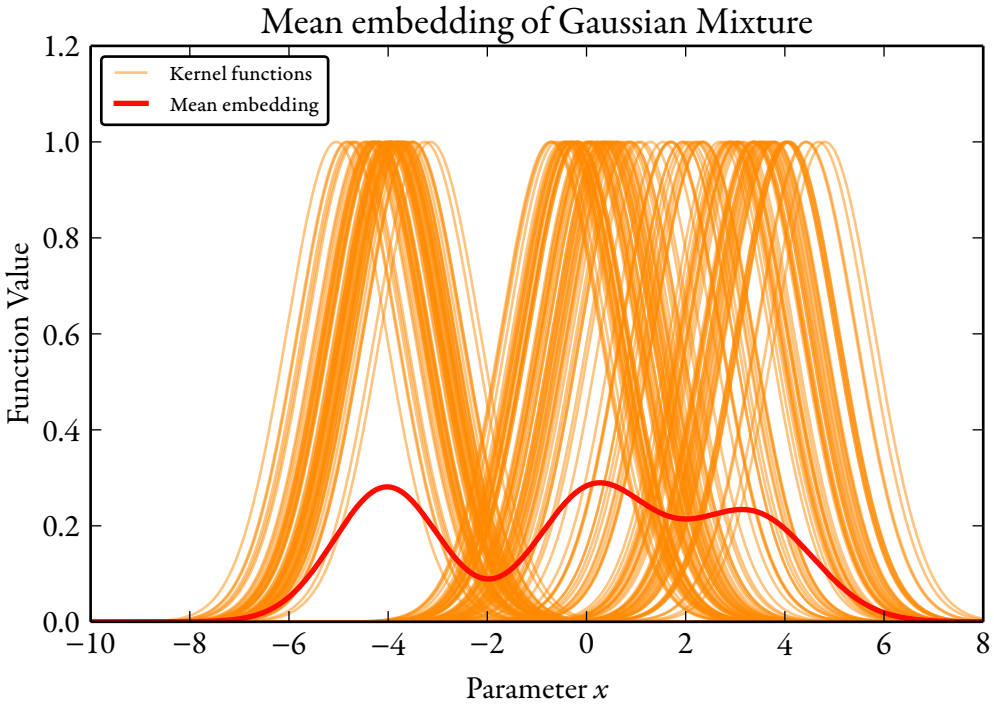


Figure 2.7.: An empirical mean embedding of a three component Gaussian mixture, created with squared exponential kernels centred at 180 sample locations.

2.7.1. Cross-Covariance Operators

Cross-covariance operators are operators between two RKHSs used to define a notion of covariance between two embedded distributions [9].

Definition 2.7.1 (Uncentred Cross-Covariance Operator)**.** Given a pair of random variables $X \in \mathcal{S}$ and $Y \in \mathcal{R}$, and associated RKHS spaces $H_X = (\mathcal{S}, k_x)$ and $H_Y = (\mathcal{R}, k_y)$, the uncentred cross-covariance operator C_{XY} is the operator from H_Y to H_X satisfying

$$C_{XY} = \mathrm{E}_{(X,Y)} \left[k_X(X, \cdot) \otimes k_Y(Y, \cdot) \right]. \tag{2.61}$$

Here $\otimes$ is the tensor product, and $k_X(X, \cdot)$ and $k_X(Y, \cdot)$ are in their role as evaluation operators. The intuition behind cross-covariance operators becomes clearer with an application of the reproducing property, which shows that for all f in H_X and g in H_Y,

$$\langle f, C_{XY} g \rangle = \mathrm{E} \left[f(X) g(Y) \right]. \tag{2.62}$$

In essence, C_{XY} measures the uncentred covariance between $f(X)$ and $g(Y)$. Viewed in a different light, the cross-covariance operator is also the mean embedding for the joint distribution $P(X, Y)$ into the RKHS $H_X \otimes H_Y$. We can see this by noting its equivalence to the mean embedding of X characterised in equation 2.50:

$$\langle g \otimes f, C_{XY} \rangle = \mathrm{E} \left[f(X) g(Y) \right]. \tag{2.63}$$

The cross-covariance operator from a space to itself C_{XX} is denoted simply as the covariance operator.

Empirical Estimation

Just like with the mean map, a converging approximation of the cross-covariance operator can be constructed from i.i.d. samples $\{(x_i, y_i)\}_{i=1}^N$ of the joint distribution $P(X, Y)$:

$$\hat{C}_{XY} = \sum_{i=1}^{N} k(x_i, \cdot) \otimes k(y_i, \cdot). \tag{2.64}$$

For characteristic kernels this estimate converges with $O(N^{-0.5})$ to the true cross-covariance operator [33].

We can derive an explicit Gram matrix expression for $\hat{C}_{XY}$ in the context of two arbitrary functions $f(X) = \mathbf{k}_X^T \boldsymbol{\alpha}$ and $g(Y) = \mathbf{k}_Y^T \boldsymbol{\beta}$ given joint samples $\{(x_i, y_i)\}_{i=1}^N$. The reproducing property requires

$$\begin{aligned} \langle f, C_{XY} g \rangle &= \mathrm{E}[f(X)g(Y)] && (2.65)\\ &\approx \sum_i f(x_i)g(y_i) && (2.66)\\ &= \sum_i \langle k(x_i, \cdot), f \rangle \langle k(y_i, \cdot), g \rangle. && (2.67)\\ & && (2.68) \end{aligned}$$

Substituting the expressions for f and g requires that

$$\hat{C}_{XY} g = \mathbf{K}_Y^T[\cdot] G_{YY} \boldsymbol{\beta}. \tag{2.69}$$

This result is quite useful in later derivations of more complex operators.

2.7.2. Conditional Probabilities

One of the fundamental operations in probability is conditioning; considering the probability of one variable when another has been fixed to a particular value. Cross-covariance operators allow for conditional distributions to be embedded in an RKHS, furthering the goal of an RKHS analogue of the laws of probability and hence of inference in these spaces.

The *conditonal embedding* $C_{Y|X}$ is an operator from H_X to H_Y, which, when acting on a point embedding $\mu[x]$ in H_X, results in the point $\mu[P(Y|X = x)]$ in H_Y [112]. In other words, $C_{Y|X}$ is the RKHS analogue of a condition distribution $P(Y|X)$.

Definition 2.7.2 (Conditional Embedding)**.** Let X and Y be random variables over $\mathbb{R}^M$, and $\mu[P(Y|X = x)]$ the mean embedding of the conditional distribution $P(Y|X = x)$. The conditional embedding $C_{Y|X}$ is an operator from H_X to H_Y, which for all $x \in \mathbb{R}^M$:

$$\mu[P(Y|X = x)] = C_{Y|X} \mu[x], \tag{2.70}$$

where $\mu[x] = \int k(x, X)\delta(x - X)\,\mathrm{d}X = k(x, \cdot)$.

The conditional embedding also obeys the reproducing property:

$$\langle g, \mu[P(Y|X = x)]\rangle = \langle g, C_{Y|X}\mu[x]\rangle \tag{2.71}$$

$$= \mathrm{E}[g(Y)|X = x]. \tag{2.72}$$

The expression for $C_{Y|X}$ is derived in [112], following from a theorem first proved in [32]. It is defined in terms of cross-covariance operators,

$$C_{Y|X} = C_{YX}C_{XX}^{-1}. \tag{2.73}$$

The full expression for a conditional embedding is then

$$\mu[P(Y|X = x)] = C_{YX}C_{XX}^{-1}\mu[x]. \tag{2.74}$$

Note that the expression C_{XX}^{-1} in this expression is not rigorous. The inverse operator does not exist in general, but regularised inverses can be computed empirically [35].

Empirical Estimate

Unfortunately, neither C_{XX} being one-to-one, nor $C_{Y|X} \in H_X$ can be assumed to hold in general [34]. However, it is possible to derive an empirical approximation $\hat{C}_{Y|X}$ under these assumptions and prove that it causes the instantiation $\hat{C}_{Y|X}\mu[x]$ to converge to $\mu[P(Y|X = x)]$.

Given samples $(x_i, y_i)_{i=1}^N$ from $P(X, Y)$, and assuming that $k(x, \cdot)$ lies in the range of C_{XX}, then the empirical estimate $\hat{C}_{Y|X}$ is given by

$$\mu[Y|X = x] = \mathbf{k}_Y[\cdot]^T(G_{XX} + N\varepsilon I)^{-1}\mathbf{k}_X[x], \tag{2.75}$$

where $\varepsilon \in \mathbb{R}$ is the regularising term, and $(\mathbf{k}_X[x])_i = k(x_i, x)$. This estimate converges to the true $\mu[P(Y|X = x)]$ at the rate $O((N\varepsilon)^{-0.5} + \varepsilon^{0.5})$ [112].

Relationship to Gaussian Process Regression

Gaussian process regression uses a set of training points $\{(x_i, y_i)\}_{i=1}^N$ to construct a distribution of functions, which, when conditioned on query variables $\{x^*\}$, produces a posterior mean function with associated covariance structure. The GP is defined through a covariance function, that also happens to be in the form of a positive-definite kernel [98].

The mean of a Gaussian process given training data $\{(x_i, y_i)\}_{i=1}^N$ and a single conditioning variable x^* is

$$\mu = \mathbf{k}_x[x^*]^T(G_{XX} + \sigma^2 I)^{-1}\mathbf{Y}, \tag{2.76}$$

where $\mathbf{Y} = (Y_1, Y_2, ...)^T$. Interestingly, this is exactly the embedding $\mu[P(Y|X = x^*)]$ from equation 2.75, assuming we have a linear kernel $k(y, y') = \mathbf{y}^T\mathbf{y}$. This relationship raises interesting questions about the link between Gaussian process regression and RKHS kernel embeddings more generally, and particularly raises possibility of assigning a physical meaning to the regularisation parameter ε, which corresponds to the process noise σ^2 in the GP case. Chapter three discusses these regularisation terms further, in the context of training procedures.

2.8. Kernel Probability Laws

The ability to embed conditional distributions has now opened the door to defining the fundamental laws of probability inside an RKHS. Just as these rules are used to derive Bayes' rule, their RKHS analogues will be used to construct its kernel equivalent and the basis for future work, kernel Bayes' rule.

2.8.1. Kernel Sum Rule

The kernel sum rule is the kernel analogue of the sum rule,

$$P(Y) = \int P(Y|X)P(X)\,\mathrm{d}X, \tag{2.77}$$

corresponding to the marginalisation of a variable X from the conditional $P(Y|X)$ through multiplication of the prior $P(X)$ and integration.

The kernel analogue of this would see a mean embedding $\mu[P(Y)] \in H_Y$ estimated from a conditional embedding operator $C_{Y|X}$ and a prior embedding $\mu[P(X)]$. It follows directly from the conditional embedding in section 2.7.2.

Theorem 2.8.1 (Kernel Sum Rule). *Given random variables X and Y, distributions $P(X)$, $P(Y)$ and $P(Y|X)$, and associated RKHSs H_X and H_Y, then $\mu[P(Y)]$ and $\mu[P(X)]$ are related by*

$$\mu[P(Y)] = C_{Y|X}\mu[P(X)], \tag{2.78}$$

$$= C_{YX}C_{XX}^{-1}\mu[P(X)]. \tag{2.79}$$

where C_{YX} and C_{XX} are cross-covariance operators [112].

Empirical Estimate

Given training samples $\{x_i, y_i\}_{i=1}^N$ from the joint, and a weighted mean embedding $\{(\alpha_i, u_i)\}_{i=1}^M$ for the prior $P(X)$, the empirical estimate of the kernel derives from a direct substitution of the empirical estimates $\mu[Y]$, $\mu[X]$ and $\hat{C}_{Y|X}$:

$$\mu[Y] = \mathbf{k}_Y^T[\cdot](G_{XX} + N\varepsilon I)^{-1}G_{XU}\boldsymbol{\alpha}, \tag{2.80}$$

where $(G_{XU})_{ij} = k(x_i, u_j)$.

2.8.2. Kernel Product Rule

The product rules relates the joint distribution $P(Y, X)$ to the conditional $P(Y|X)$ and the marginal $P(X)$ by $P(Y, X) = P(Y|X)P(X)$. The kernel analogue of this rule relates the associated mean embeddings of these distributions.

Theorem 2.8.2 (Kernel Product Rule). *Given random variables X and Y, distributions $P(X)$, $P(Y)$ and $P(Y, X)$, and associated RKHSs H_X, H_Y and $H_X \otimes H_Y$ then $\mu[P(X, Y)]$*

and $\mu[P(X)]$ *are related by*

$$\mu[P(X, Y)] = C_{(YX)X} C_{XX}^{-1} \mu[P(X)], \tag{2.81}$$

where $C_{(YX)X}$ *is the cross-covariance operator from* $\mathcal{H}_X$ *to joint space* $\mathcal{H}_Y \otimes \mathcal{H}_X$ *[34].*

A simple intuition for the kernel analogue of this rule was given in [31]. Let $Z = (X, Y)$. Then we can write $P(Y, X)$ in terms of the sum rule by noting that

$$P(Z|X) = P(X, Y|X')\delta(X - X'), \tag{2.82}$$

and hence, that

$$P(Z) = \int P(Z|X)P(X)dX. \tag{2.83}$$

Direct application of the kernel sum rule for this expression yields

$$\mu[P(X, Y)] = C_{(YX)X} C_{XX}^{-1} \mu[P(X)]. \tag{2.84}$$

Empirical Estimate

The empirical estimate also follows from kernel sum rule. Given training samples $\{x_i, y_i\}_{i=1}^N$ from the joint, and a weighted mean embedding $\{(\alpha_i, u_i)\}_{i=1}^M$ for the prior $P(X)$, then

$$\mu[P(X, Y)] = \mathbf{k}_{XY}^T[\cdot, \cdot](G_{XX} + N\varepsilon I)^{-1} G_{XU}\boldsymbol{\alpha}, \tag{2.85}$$

where $(\mathbf{k}_{XY}^T[\cdot, \cdot])_i = k_X(x_i, \cdot) \otimes k_Y(y_i, \cdot)$. This is identical to the expression for the weights of $P(Y)$ in section 2.8.1, only with a basis in the joint space.

2.9. Kernel Bayes' Rule

Finally, with the kernel analogues of the fundamental laws of probability, we are able to derive an expression that applies Bayes' theorem to empirical estimates of mean-map embeddings of probability distributions into an RKHS— otherwise known as kernel Bayes'

rule [34, 35].

Given a prior embedding $\mu[P(X)]$ and a likelihood $\mu[P(Y|X)]$, kernel Bayes' rule computes the posterior embedding $\mu[P(X|Y = y)]$ conditioned on a given observation y. From the conditional embeddings in equation 2.74:

$$\mu[P(X|Y = y)] = C_{X|Y}\mu[y] \tag{2.86}$$

$$\mu[P(X|Y = y)] = C_{XY}C_{YY}^{-1}\mu[y]. \tag{2.87}$$

The equivalence between a cross-covariance operator from $H_X \to H_Y$ and the mean map in the product space $H_X \otimes H_Y$ described in section 2.7.1 means we can use joint embeddings to derive expressions for C_{YY} and C_{XY} above.

In particular, we equate C_{XY} with the joint embedding $\mu[P(X, Y)]$ from equation 2.81:

$$C_{XY} = \mu[P(X, Y)] \tag{2.88}$$

$$= C_{(YX)X}C_{XX}^{-1}\mu[P(X)]. \tag{2.89}$$

The cross-covariance operator C_{YY} is similarly defined through the kernel product rule, this time using the product space embedding $\mu[P(Y, Y)]$:

$$C_{YY} = C_{(YY)X}C_{XX}^{-1}\mu[P(X)]. \tag{2.90}$$

Substituting these operators into equation 2.86 yields kernel Bayes' rule:

Theorem 2.9.1 (Kernel Bayes' Rule). *Let X and Y be random variables with distribution with prior $P(X)$ and joint $P(X, Y)$, and let y be an observation of Y. Then,*

$$\mu[P(X|Y = y)] = C_{(YX)X}C_{XX}^{-1}\mu[P(X)]\left[C_{(YY)X}C_{XX}^{-1}\mu[P(X)]\right]^{-1}\mu[y], \tag{2.91}$$

where $C_{(YX)X}$, C_{XX} and $C_{(YY)X}$ are the associated cross-covariance operators [34, 35].

2.9.1. Empirical Estimate

From the empirical kernel probability laws, we can derive an expression for evaluating KBR using a set of samples $\{x_i, y_i\}_{i=1}^N$ from the joint $P(X, Y)$ and $\{u_j\}_{j=1}^M$ from the prior $P(X)$.

In [34], these prior samples could also have real (and possibly negative) weights $\{\gamma_j\}_{j=1}^M$.

Next are the empirical estimates of operators C_{XY} and C_{YY} in equations 2.88 and 2.90. These estimates are of the form

$$\hat{C}_{XY} = \sum_{i=1}^{N} \beta_i k(x_i, \cdot) \otimes k(y_i, \cdot), \tag{2.92}$$

$$\hat{C}_{YY} = \sum_{i=1}^{N} \beta_i k(y_i, \cdot) \otimes k(y_i, \cdot), \tag{2.93}$$

with common weights $\boldsymbol{\beta}$ as demonstrated by equation 2.85:

$$\boldsymbol{\beta} = (G_{XX} + N\varepsilon I)^{-1} G_{XU} \boldsymbol{\gamma}. \tag{2.94}$$

The final expression for the KBR requires a different type of regularised inverse for $\hat{C}_{YY}$ than was used for $\hat{C}_{XX}$. This is because the Gram matrix expression may contain negative eigenvalues, and hence must first be squared in order to ensure the operator remains positive-definite:

$$\mu[X|Y = y] = \hat{C}_{XY}(\hat{C}_{YY}^2 + \delta I)^{-1} \hat{C}_{YY} \mu[y]. \tag{2.95}$$

The Gram matrix expression that follows is

$$\mu[X|Y = y] = \mathbf{k}_X^T[\cdot] R_{X|Y} \mathbf{k}_Y[y], \tag{2.96}$$

where

$$R_{X|Y} = D(\beta) G_{YY} \left((D(\beta) G_{YY})^2 + \delta I \right)^{-1} D(\beta), \tag{2.97}$$

$D(\beta) = \text{Diag}(\beta)$, and $(\mathbf{k}_Y[y])_i = k(y_i, y)$. The convergence properties of equation 2.96 are discussed in detail in [34] and [35]. Under normal conditions, assuming the prior is sampled equally often as the joint, convergence follows a $O(n^{4/27})$ rate, which while slow, is a lower bound that is exceeded in practice.

Finally, we have an expression to compute Bayes' rule inside an RKHS. Figure 2.8 gives

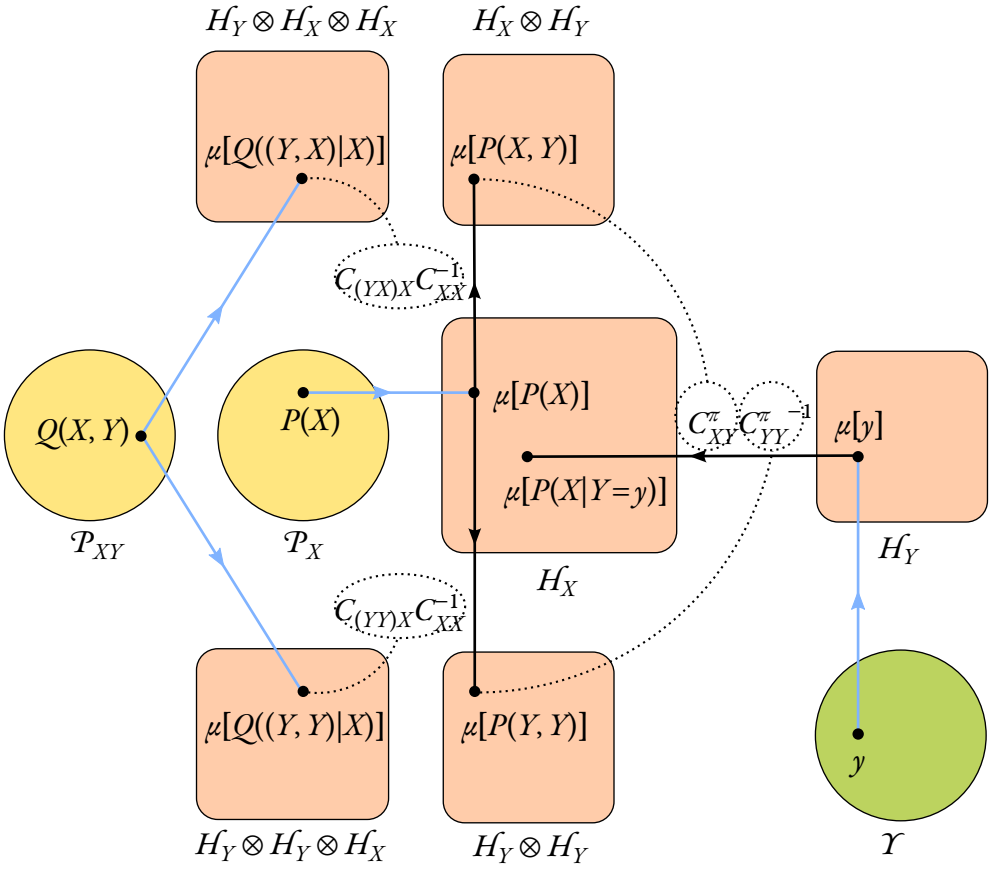


Figure 2.8.: Diagram of the operations involved in computing the posterior embedding $\mu[P(X|Y=y)]$ in KBR. Beginning with the 'input data' of the algorithm, we have the distributions $Q(X,Y)$ (the joint), $P(X)$ (the prior), and an observation y of Y. The distributions live inside the yellow spaces $\mathcal{P}_{XY}$ and $\mathcal{P}_X$ respectively, whilst the variable y is from the green space $\mathcal{Y}$, for instance $\mathbb{R}^N$. The mean map, indicated by the blue arrows, is used to map these objects into a number of different Hilbert spaces. $P(X)$ is mapped to the prior embedding $\mu[P(X)] \in H_X$, whilst y is mapped to $\mu[y] \in H_Y$. Kernel Bayes' rule itself is the operator $C_{XY}^{\pi}C_{YY}^{\pi\,-1}$ which maps this embedding $\mu[y]$ to the posterior embedding $\mu[P(X|Y=y)] \in H_x$, indicated by the solid black leftward arrow. These two operators can themselves be seen as elements of RKHSs, indicated by the dotted lines equating C_{XY}^{π} with $\mu[P(X,Y)] \in H_X \otimes H_Y$ and C_{YY}^{π} with $\mu[P(Y,Y)] \in H_Y \otimes H_Y$. These two elements are mapped from H_X by two other operators which are also elements of the two left-most RKHSs, constructed through the mean map on $Q(X,Y)$.

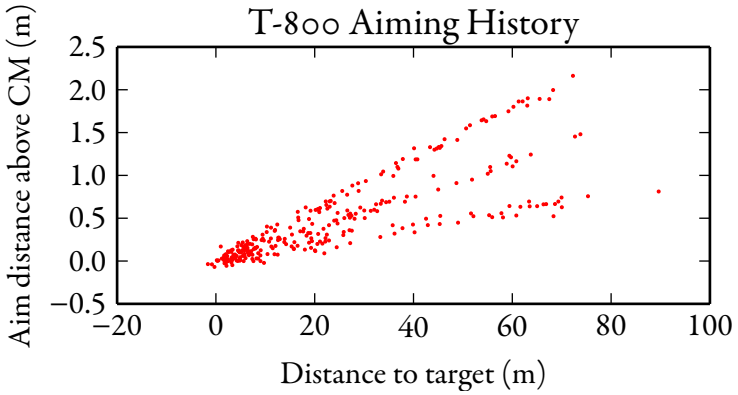
a visual overview of the KBR, illustrating the embedded estimates of various distributions, and the computation of the posterior embedding $\mu[P(X|Y=y)]$.

This algorithm forms the basis for our contributions in further chapters. Note the immediate issue: we have derived an expression for $\mu[P(X|Y=y)] \in H_X$, but have so far not discussed estimating $P(X|Y=y)$ itself. Efficient and accurate recovery of the distribution and associated information is a non-trivial problem, to which we devote our attention in chapter three.

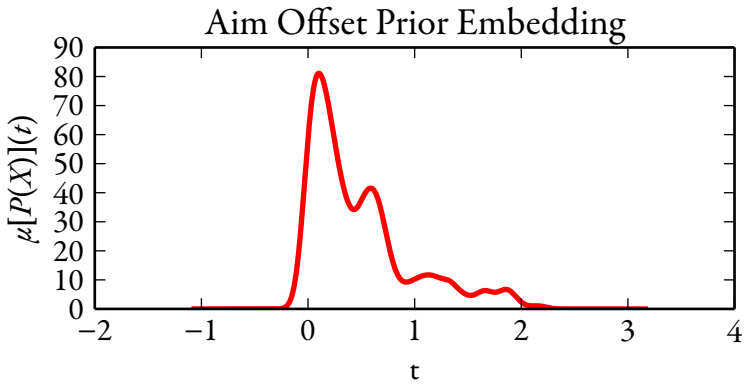
2.9.2. Example

The T-800 model robot has just located subject John Connor, standing across an empty street. The robot must now initiate Greeting Protocols using its M-27 phased-plasma pulse rifle. Using a built-in laser range-finder, the robot estimates the distance to the subject at 53.68 metres. At this distance, how far above John Connor's centre of mass should the robot aim to achieve maximally efficient greeting?

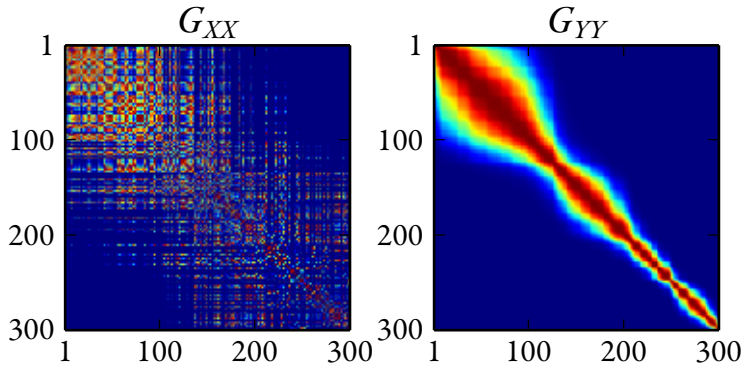
Fortunately, the robot has collected the aim offsets from 300 previous successful implementations of the Greeting Protocol at various distances from the subject, and so decides to use kernel Bayes' rule to compute the posterior probabilities of aim offsets, conditioned on its measurement of distance to subject.



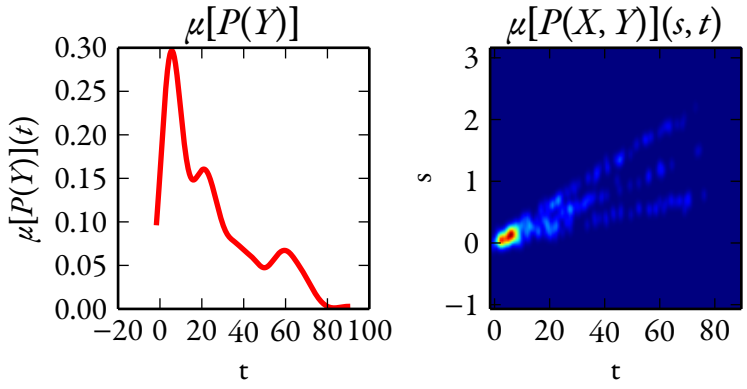
Let X represent the distance above the subject's centre of mass (CM), and Y be the distance to the subject. The robot opts to use the X data as samples from the prior $P(X)$ and the (X, Y) data as samples from the joint $P(X, Y)$. The robot selects a Gaussian kernel to perform the inference, with bandwidths $\sigma_x = 3.2$, and $\sigma_y = 0.075$. The prior embedding $\mu[P(X)] = \sum_{i=1}^{300} k_X(x_i, \cdot)$ is then computed with this kernel:



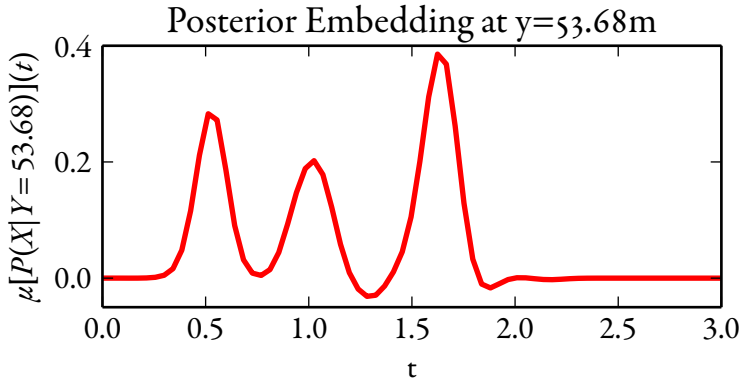
Next, the robot computes the Gram matrices G_{XX} and G_{YY} required to estimate the conditional operators, with $(G_{XX})_{ij} = k_X(x_i, x_j)$ and $(G_{YY})_{ij} = k_X(y_i, y_j)$.



Because the robot chose to use the X values of the joint as the prior U, the prior weights γ are all equal to $\frac{1}{300}$. Therefore, with suitable choice of regulariser $\varepsilon = 0.006$, it can now compute β from equation 2.94. The weights beta are shared by the embedding $\mu[P(Y)] = \sum_{i=1}^{300} \beta_i k_Y(y_i, \cdot)$, shown below-left, and $\mu[P(X, Y)] = \sum_{i=1}^{300} \beta_i k_Y(y_i, \cdot) k_X(x_i, \cdot)$ shown below-right.



Finally, the robot computes the weights $\boldsymbol{\alpha}$ of the posterior embedding $\mu[P(X|Y=y)]$ by $\boldsymbol{\alpha} = R_{X|Y}\mathbf{k}_Y[y]$, with $R_{X|Y}$ from equation 2.97 (using $\delta = 0.8$) and the vector instantiating its observation $(\mathbf{k}_Y(y))_i = k_Y(y_i, 53.68)$. The resultant embedding is $\mu[P(X|Y=20.12)] = \sum_{i=1}^{300} \alpha_i k_X(x_i, \cdot)$.



Unfortunately, the T-800 has now hit a critical problem. It cannot interpret this embedding as a probability distribution, noting that it drops below zero at some points. As the robot neglected to parse the next chapter of this thesis, it cannot recover a true probability distribution from the embedding, and falters long enough for John Connor to escape.

2.10. Summary

In this chapter we introduced kernel Bayes' rule, and provided the required background in functional analysis and reproducing kernel Hilbert spaces. The key idea of the chapter is

that probability distribution can be represented as elements in a type of vector space called a reproducing kernel Hilbert space. Critically, it is possible to create such representations with only samples from the distribution, rather than an explicit parametric form.

These embeddings are able to be manipulated through operators which act analogously to the sum and product rules of probability. By embedding prior and likelihood distributions, a combination of these operators can be applied to perform Bayes' rule inside the reproducing kernel Hilbert space. The result is an element of the space corresponding to the posterior distribution that would have been obtained had Bayes rule been applied directly to the prior and likelihood. However, kernel Bayes' rule avoids the need to compute the costly marginalisation integral that would be otherwise required.

Chapter 3

Multi-modal Regression and Filtering

Multi-modal probability distributions appear in a wide variety of inference contexts, especially in machine learning and robotics. In this chapter we present extensions to kernel Bayes' rule to allow multi-modal and multi-dimensional posterior density estimation, for both regression and filtering problems. We apply these techniques to a variety of multi-modal problems in robotic motion estimation.

3.1. Motivation

Kernel Bayes' rule's ability to learn the prior and likelihood from samples, without restriction to a particular parametric form is a useful attribute in disciplines such as robotics and machine learning where parametric models can be unknown or difficult to estimate. However, the output of kernal Bayes' rule (KBR) is a mean-map embedding in a reproducing kernel Hilbert space (RKHS), not a probability distribution. The original methods of obtaining posterior estimates were either maximum *a-posteriori* (MAP) points, or expectations of functions with respect to the posterior [31]. A technique to estimate the full posterior density function would enable us to quantify the uncertainty associated with a prediction, something critical to any automated decision making, and also to use the posterior as a prior distribution in subsequent analysis with a different algorithm. Such an algorithm could potentially see wide application in machine learning, perhaps replacing the ubiquitous Gaussian process regression in circumstances where the Gaussianity

assumptions of this algorithm do not apply.

Another area where such a fully non-parametric, multi-modal method for Bayesian inference would be valuable is the time-recursive Bayesian inference or filtering commonly used in robotics. Current filtering methods are either restricted to particular distributions or else are unable to learn complex multi-dimensional dynamics. A Bayesian filter capable of modelling both non-linear dynamics and multi-modal distributions would therefore be particularly desirable in filtering applications where complex, multi-modal distributions arise. Examples include bearing-only tracking, multi-hypothesis tracking, or in cases of poorly known dynamics. In principle, a SLAM implementation could also be developed which included probabilistic data association as an inherent feature. A non-linear multi-modal filter would also allow for multiple motion models to be considered simultaneously, but with a fully Bayesian treatment of the resultant distribution. This is in contrast to the filter-bank methods commonly used.

3.2. Contributions

This chapter focuses on the problem of extracting posterior estimates from kernel Bayes' rule, and hence on the difficulty of performing multi-modal non-parametric inference. Such problems often arise in the context of robotics motion estimation, owing to complex motion models without parametric forms and noisy sensors. To address these problems, we present the following contributions:

A novel mixture distribution pre-image algorithm to obtain a full posterior estimate from the KBR algorithm. This extends the KBR to perform not only non-linear and non-parametric, but also multi-modal inference. Our technique is well-suited to both regression and filtering applications of KBR, as recursive inference can happen purely within the RKHS and a distribution estimate recovered only when required. We denote our extension as multi-modal kernel Bayes' rule, or MKBR. Finding these estimates of the posterior with KBR involves solving the pre-image problem; approximating the inverse of mapping between a set and an RKHS. We present two techniques for estimating the pre-image of the embedded posterior. The first relies on embedding a mixture distribution into the RKHS and minimising the distance between this distribution and the embedded poste-

rior. The second involves truncating and normalising the weights of the embedding to directly estimate a posterior.

A new training scheme necessary for automatic parameter estimation. In both the regression and filtering cases, learning the free parameters in the algorithm requires a cost function that properly accounts for the multi-modality of the output of our algorithm. We present a training system utilising the negative log-probability of a leave-out set of the training data, that will also be useful in later chapters.

Novel applications of our multi-modal inference system to difficult motion estimation problems in robotics. We first illustrate our techniques with an experiment tracking a simulated particle moving through a set of randomly chosen trajectories. This filtering problem is multi-modal and has multi-dimensional input and output, and the MKBR algorithm is able to demonstrate learning the hidden variable controlling which path the particle takes. We then consider an experiment tracking a slot car moving around a track from inertial data taken from a small on-board inertial measurement unit (IMU). This experiment demonstrates learning complex dynamic models without parametric forms and in the presence of noise.

Significant extension of a robotics algorithm for indoor motion planning presented in [88]. The algorithm builds a normalised probabilistic velocity map for robot path planning based on the recorded trajectories of pedestrians in the area. The original algorithm relied on Gaussian process (GP) regression to infer this distribution, but the uni-modal estimates from a GP cause the algorithm to break down when pedestrians move in different directions through the same area. Our MKBR extension allows for multi-modal distributions of direction, and so much more reliable path planning.

3.3. Related Work

Filtering is a fundamental problem in robotics, in which a probabilistic representation of the state of the world or the robot itself is periodically updated with new measurements from sensors. The venerable Kalman filter remains a popular method to implement filtering [58], but contains strong assumptions about the prediction and observation models, and the underlying distributions. All models must be linear, and all variables are assumed

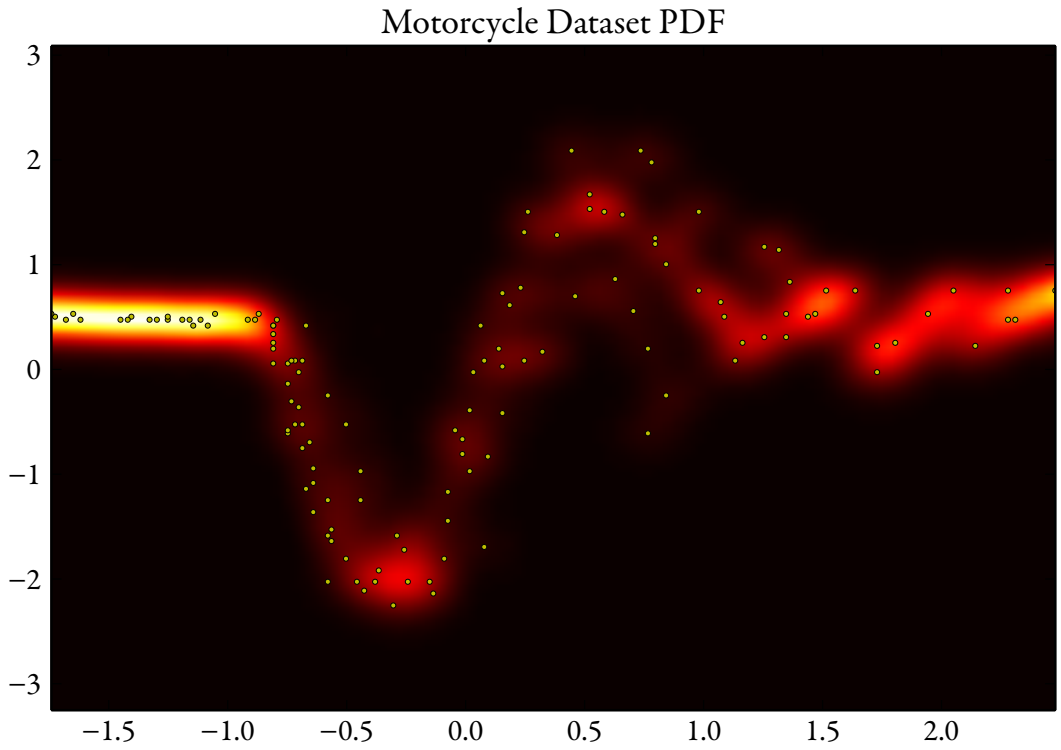


Figure 3.1.: Illustration of the multi-modal (and heteroscedastic) output of our MKBR algorithm as presented in this chapter, applied to the well-known 'motorcycle' dataset.

to be normally distributed (Gaussian). In situations where these assumptions hold, the Kalman filter has demonstrated many years of successful application.

The extended Kalman filter (EKF) relaxed the need for linear models by linearising the prediction and observation update models at the current mean and covariance estimate [53]. However, if the underlying models are strongly non-linear or the filter is poorly initialised, numerical stability is difficult to achieve [55].

The unscented Kalman filter (UKF) was made partly to overcome these issues of instability. It uses an alternative linearisation approach in which deterministic samples of the random variables are propagated through the non-linear models. The image of these samples under the model is then used to estimate the Gaussian for the next state [56]. This filter has demonstrated greater numerical accuracy and tends to be simpler to implement than the EKF, and has also been widely adopted when the Gaussian assumption is valid.

A number of filtering algorithms have attempted to also relax the Gaussianity assumption placed on the underyling distributions. These filters operate on representations of distributions formed through the sum of simpler basis functions.

The earliest examples of these filters were Parzen density and mixture model filters, which tile basis functions such as Gaussians or sawteeth in order to represent distributions [106, 3], but are limited to linear models and suffer from poor dimensional scaling in practice.

Particle filters are a very general filter developed to represent arbitrary distributions and non-linear models, by approximating distributions with a set of point samples ('particles') that are mapped through the transition and observation models. However, in general these methods suffer from very poor dimensional scaling and difficulties with sample degeneracy or impoverishment [5, 40]. Progress has been made to overcome these problems beyond the standard sequential importance sampling particle filters; EKF and UKF particle filters use those filters to approximate the proposal distribution, reducing sample impoverishment but introducing linearisation errors [28, 81]. Rao-Blackwellised particle filters demonstrate better dimensional scaling [23] but require structural knowledge of posterior distributions. Gaussian particle filters also scale more favourably with dimension but are restricted to Gaussian posteriors [69]. Gaussian sum filters relax this restriction to representation as a Gaussian sum [70] but suffer from mixture component impoverishment. Both algorithms are also sensitive to linearisation errors.

All of the filters described above require analytical expression for the transition and observation models, whether linear or non-linear. Few filters thus far have been extended to enable learning of prediction and observation models from data. The noise parameters of the EKF were learned in [1]. The GP-BayesFilter particle filter uses a standard sequential importance re-sampling (SIR) particle filter but learns the state transition and observation models from data using a set of Gaussian processes [65].

Apart from their use in filtering, GPs [98] themselves are a popular tool in robotics for non-parametric regression when the assumptions of Gaussian priors, likelihoods and posteriors are valid. GPs perform inference over a space of smooth functions, any finite sampling from which is Gaussian-distributed. As such they are capable of modelling complex relationships between variables with Gaussian marginals, and have been used for modelling terrain and occupancy [121, 87], optimal control [22], and for planning based on information gain [107]. GP mixture models relax the Gaussianity assumption of the posterior but retain a strong assumption of mixture component independence [120].

One class of filtering and regression techniques which relax the Gaussianity assumption, but retain non-parametric representations and high-dimensional scaling, are those based on representing distributions as elements of a Hilbert space of functions. These methods have shown a number of advantages over traditional estimation techniques, especially with complex, high-dimensional distributions when very little prior information is known. Methods based on orthogonal function bases and have good sparsity and scaling properties in theory, but the filtering equations are generally insoluble [16, 79]. Methods using reproducing kernel Hilbert spaces have had more success, initially with a Bayesian filter approximated heuristically [112], assuming additive contributions from the observation and state transition models. An estimate to the filtering problem posed in the form of a hidden Markov model was recently developed [111] that can learn complex non-linear models, but is limited to producing a maximum *a-posteriori* estimate.

The kernel Bayes' rule algorithm, used as the basis for our approach to regression and filtering [34] in this chapter, provides a converging estimate to full Bayesian inference. It learns non-linear models from training data, has no restrictions on the shape of prior or posterior distributions, and has demonstrated scalability to high dimension. Its main limitation is that it recovers only a maximum *a-posteriori* estimate at each time-step. There is no estimation of the uncertainty in the result. We will rectify this limitation by solving the

pre-image problem for the KBR posterior, and in doing so, recover an estimate of the full posterior density function. Earlier versions of some of our results in this chapter appeared in [80]. However, we have since modified both our training algorithm and our posterior estimation scheme, and also performed more extensive experiments.

However, before describing our algorithm, we first give a brief description of the filtering variant of kernel Bayes' rule from [34]. This will enable us to demonstrate posterior recovery on both filtering and regression problems in later sections.

3.4. Kernel Bayes' Rule Filtering

Bayesian filtering is a special case of the inference problem in which we attempt to estimate a dynamic state X_t over time using a set of noisy measurements of that state $Y_1, \ldots, Y_t$. By making a Markov assumption, the problem can be framed in terms of two stochastic functions; a state transition model $s: X_t \rightarrow X_{t+1}$ that encapsulates how the state evolves with time, and an observation model $o: X_t \rightarrow Y_t$ that relates the state to an observation. Probabilistically, the state transition model is encoded in the distribution $P(X_{t+1}|X_t)$, and the observation model in $P(Y_{t+1}|X_{t+1})$.

The goal of Bayesian filtering is to estimate the state X at time $t+1$ in terms of these distributions and our estimate of X at the previous time step t. Bayes' theorem relates these various models by

$$P(X_{t+1}|Y_{1:t+1}) = \frac{P(Y_{t+1}|X_{t+1})P(X_{t+1}|Y_{1:t})}{P(Y_{t+1}|Y_{1:t})}. \tag{3.1}$$

The KBR analogue of this Bayesian filtering equation uses a set of samples $\{(x_{t'}, y_{t'})\}_{t'=0}^{N}$ assumed to be from the joint distribution $P(X, Y)$, and uses them to learn the state transition and observation models. Given an embedded estimate of the state at some time t, $\mu[P(X_t|Y_{1:t}=y_{1:t})]$ that is based on previous observations $y_{1:t}$, and an observation y_{t+1}, the KBR filter computes the embedded estimate $\mu[P(X_{t+1}|Y_{1:t+1}=y_{1:t})]$ at time $t+1$. For brevity we will write $\mu[P(X_t|Y_{1:t}=y_{1:t})]$ as $\mu[P(X_t|y_{1:t})]$.

To write equation 3.1 in terms of kernel embeddings, we first note that $P(X_{t+1}|Y_{1:t})$ is

the marginal of the transition model with respect to the prior:

$$P(X_{t+1}|Y_{1:t}) = \int P(X_{t+1}|X_t)P(X_t|Y_{1:t})\,\mathrm{d}X_t. \tag{3.2}$$

Using the kernel sum rule in equation 2.78, we can write the corresponding embedding as

$$\mu[P(X_{t+1}|Y_{1:t})] = C_{X_{+1}X}C_{XX}^{-1}\mu[P(X_t|Y_{1:t})], \tag{3.3}$$

where $C_{X_{+1}X}$ is the cross-covariance operator between states at $0, \ldots, t$ and $1, \ldots, t+1$. Decomposing the denominator $P(Y_{t+1}|Y_{1:t})$ with the sum rule yields

$$P(Y_{t+1}|Y_{1:t}) = \int P(Y_{t+1}|X_{t+1})P(X_{t+1}|Y_{1:t})\,\mathrm{d}X_{t+1}, \tag{3.4}$$

for which the kernel sum rule again provides an expression,

$$\mu[P(Y_{t+1}|Y_{1:t})] = C_{YX}C_{XX}^{-1}\mu[P(X_{t+1}|Y_{1:t})]. \tag{3.5}$$

Substituting equations 3.3 and 3.5 gives an expression for $\mu[P(Y_{t+1}|Y_{1:t})]$ in terms of known quantities:

$$\mu[P(Y_{t+1}|Y_{1:t})] = C_{YX}C_{XX}^{-1}C_{X_{+1}X}C_{XX}^{-1}\mu[P(X_t|Y_{1:t})]. \tag{3.6}$$

From this point, direct application of kernel Bayes' rule using prior $P(X_{t+1}|Y_{1:t})$ and likelihood $P(Y_{t+1}|X_{t+1})$ yields an expression for the posterior

$$\mu[P(X_{t+1}|y_{1:t+1})] = C_{XY}^{t+1}{C_{YY}^{t+1}}^{-1}\mu[y_{t+1}] \tag{3.7}$$

with $C_{XY}^{t+1} = C_{(YX)X}C_{XX}^{-1}\mu[P(X_{t+1}]|y_{1:t}]$ (equation 2.88) and C_{YY}^{t+1} associated with the corresponding embedding in equation 3.6.

3.4.1. Empirical Estimate

Given a set of samples $\{x_t, y_t\}_{t=1}^N$ from the stochastic process at times $t = 1 \ldots N$, the empirical embedding of $\mu[P(X_t|y_{1:t})]$ of the probability of X at time t given observations $y_1 \ldots y_t$

is of the form

$$\mu[X_t|y_{1:t}] = \mathbf{k}_X[\cdot]^T\boldsymbol{\alpha}, \tag{3.8}$$

where $(\mathbf{k}_X[\cdot])_i = k(x_i, \cdot)$, and $\boldsymbol{\alpha}$ is a vector of real weights. This forms the prior for the filtering step. The empirical estimate of the prediction step that transforms this into an embedding $\mu[P(X_{t+1}|y_{1:t})]$ has the expression

$$\mu[X_{t+1}|Y_{1:t}=y_{1:t}] = \mathbf{k}_X[\cdot]^T\boldsymbol{\beta}, \tag{3.9}$$

$$\boldsymbol{\beta} = (G_{XX} + N\varepsilon I)^{-1}G_{XX_{+1}}(G_{XX} + N\varepsilon I)^{-1}G_{XX}\boldsymbol{\alpha} \tag{3.10}$$

where ε is a regularisation parameter, and $G_{XX_{+1}}$ is the Gram matrix between the states $X_0, \dots, X_t$ and $X_1, \dots, X_{t+1}$. The observation update for the posterior distribution embedding $\mu[P(X_{t+1}|y_{1:t+1})]$ given an observation y_{t+1} is identical to the KBR conditioning step in equation 2.96:

$$\mu[X_{t+1}|y_{1:t+1}] = \mathbf{k}_X[\cdot]^T D(\boldsymbol{\beta})G_{YY}((D(\boldsymbol{\beta})G_{YY})^2 + \delta I))^{-1}D(\boldsymbol{\beta})\mathbf{k}_Y[y_{t+1}], \tag{3.11}$$

where $D(\boldsymbol{\beta}) = \text{Diag}(\boldsymbol{\beta})$ and δ is a regularisation parameter.

If we have weighted samples from a prior distribution at $t = 1$, $\{u_i, \gamma_i\}_{i=1}^M$ with u drawn from the domain of X, and an initial observation y_1, then the KBR can be applied directly to determine an initial embedding for the filter at $t = 1$. Otherwise, [34] suggests the conditional embedding $\mu[P(X_1|y_1)]$, given by

$$\mu[P(X_1|y_1)] = \mathbf{k}_X[\cdot]^T(G_{YY} + N\varepsilon I)^{-1}\mathbf{k}_Y[y_1], \tag{3.12}$$

from equation 2.74.

3.5. Kernel Bayes' Rule Posterior Recovery

Computing an estimate of the posterior distribution from either the regression or filtering variants of the KBR is vital to analysing complex, multi-modal posterior distributions, or using these posteriors as priors in other algorithms. Unfortunately, there is no analytical

method for computing the inverse of a mean embedding, in fact the inverse may not even be well-defined in general [108]. Finding (possibly approximate) inverse mappings from an RKHS embedding to the original space is known as the pre-image problem, and if we would like to obtain posterior estimates from KBR, it is the problem that needs to be solved.

3.5.1. The Pre-image Problem

Kernel algorithms for machine learning work by creating a mapping between an arbitrary set $\mathcal{S}$ and an often higher-dimensional feature space, in which a richer set of mathematical tools can be applied. In the case of KBR, the set $\mathcal{S}$ is the space of probability distributions on a random variable X, $\mathcal{P}_X$. These points are mapped to a reproducing kernel Hilbert space H, through the mean map $\mu[\cdot]\colon \mathcal{P}_X \to H$. Inference is performed in H, with the result being a posterior point $f \in H$. This point converges to the mean map of the true posterior $\mu[P(X|Y=y)]$ as the number of training points goes to infinity.

However, to recover an estimate of the posterior itself, we need to compute the pre-image of f under $\mu[\cdot]$, or in other words, the distribution $Q \in \mathcal{P}_X$ such that $\mu[Q] = f$. An example of this pre-image problem for KBR and a possible solution is illustrated in figure 3.2.

Unfortunately, the mean mapping μ is not guaranteed to be one-to-one, and the inverse $\mu^{-1}[v]$ may not even exist for some points $v \in H$. What we can do is search for points in $\mathcal{P}_X$ which, when mapped into H, are nearby f in terms of the Hilbert norm. We can then use such a point as an approximate pre-image [105]. In other words, given a point $f \in H$, we would like to find the point p^* defined such that

$$p^* = \operatorname*{argmin}_{p \in \mathcal{P}_X} \|\mu[p] - f\|_H. \tag{3.13}$$

Figure 3.3 illustrates this idea. Although the goal is a point in $\mathcal{S}$, forward embeddings only are utilised to test candidate points, with their performance measured by the RKHS distance between their embedding and the actual embedding.

Unfortunately, exploring the entire space of probability distributions to find the distri-

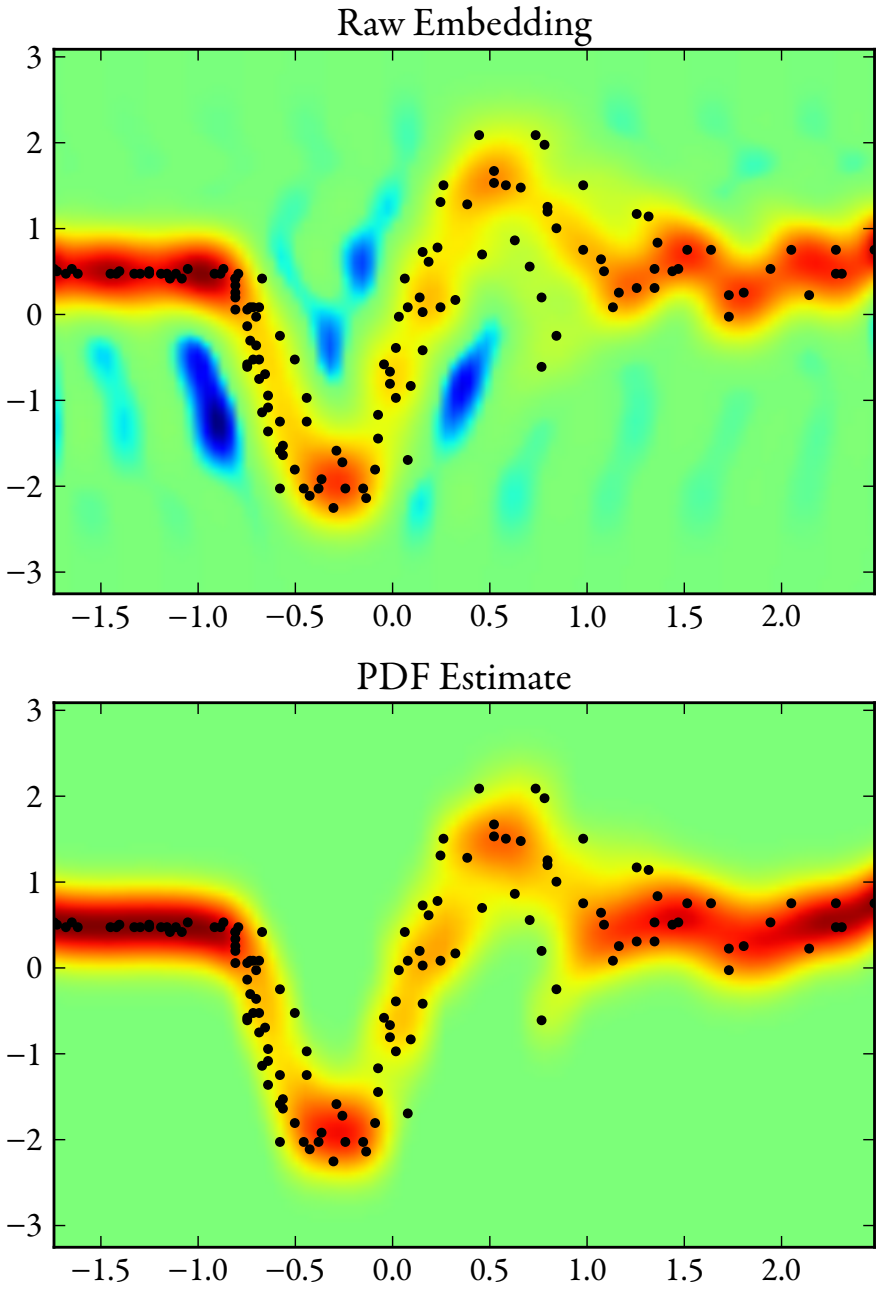


Figure 3.2.: (Top) The raw posterior embedding, with blue regions as negative values. (Bottom) PDF estimate using Gaussian mixture weights centered on the training points.

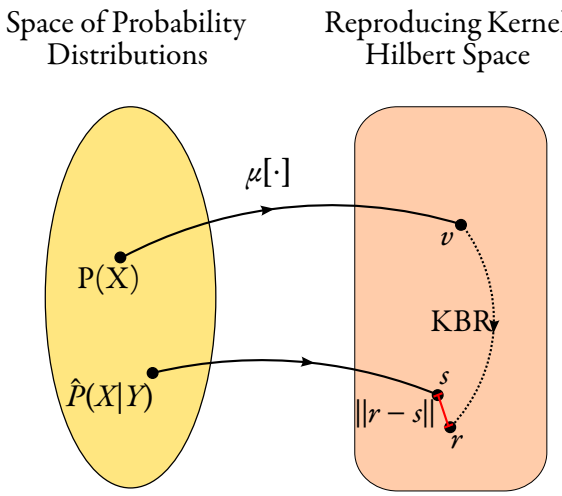


Figure 3.3.: An illustration of the pre-image problem in KBR. The posterior distribution is a point r in the RKHS. We try to find a distribution $\hat{P}$, which, when mapped into the RKHS to a point s, is close to the posterior point r.

bution embedding $\mu[\hat{P}]$ closest to the posterior is intractable. To reduce the computational burden to realistic levels, we need to make additional approximations or assumptions. In the next sections we propose two approximation methods that enable tractable posterior recovery from KBR.

3.5.2. Fixed Kernel Mixture Pre-image Method

One method for posterior recovery involves assuming a particular parametric form which facilitates the inverse mapping (in this case a mixture of kernel functions), and then finding an instance of that form closest to the true posterior in the Hilbert space metric. This corresponds to restricting the search space for our pre-image to a subset of distributions $\mathcal{Q}_X \subset \mathcal{P}_X$.

For a random variable X, let P be the true posterior, and $\hat{P}$ our recoverable estimate. P and $\hat{P}$ are elements of the set of all distributions on X, $\mathcal{P}_X$. The problem is to determine the $\hat{P}$ which has an embedding closest to the embedding of P:

$$\hat{P}^* = \operatorname*{argmin}_{\hat{P}\in\mathcal{P}_X} \|\mu[\hat{P}] - \mu[P]\|^2 \tag{3.14}$$

$$= \operatorname*{argmin}_{\hat{P}\in\mathcal{P}_X} \left(\langle\mu[P],\mu[P]\rangle - 2\langle\mu[\hat{P}],\mu[P]\rangle + \langle\mu[\hat{P}],\mu[\hat{P}]\rangle\right) \tag{3.15}$$

$$= \operatorname*{argmin}_{\hat{P}\in\mathcal{P}_X} \frac{1}{2}\langle\mu[\hat{P}],\mu[\hat{P}]\rangle - \langle\mu[\hat{P}],\mu[P]\rangle. \tag{3.16}$$

One choice for the search space is the space of distributions parametrised by weighted mixtures of fixed component distributions $\hat{P}_i$. This restricted space has the advantage of making the optimisation in equation 3.14 convex [108].

For a set of components $\{\hat{P}_i\}_{i=1}^M$ weighted by a vector $\mathbf{w}$, distributions in $\mathcal{Q}_0$ take the form

$$\hat{P}_{\mathbf{w}} = \sum_{i=0}^{M} w_i \hat{P}_i, \;\; \text{s.t} \sum_{i=0}^{M} w_i = 1, w_i > 0 \;\; \forall i. \tag{3.17}$$

Now, assume $\mu[P]$ is the posterior embedding of KBR given a set of joint samples $\{(x_i, y_i)\}_{i=1}^N$ and so let $\mu[P] = \mathbf{k}_X[\cdot]^T\boldsymbol{\alpha}$, with $\boldsymbol{\alpha} \in \mathbb{R}^N$. Using the parametrisation in equation 3.14 yields

$$P_w^* = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^N} \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} w_i w_j \langle\mu[\hat{P}_i],\mu[\hat{P}_j]\rangle - \sum_{i=1}^{M}\sum_{j=1}^{N} w_i\alpha_j\langle\mu[\hat{P}_i], k(x_j,\cdot)\rangle. \tag{3.18}$$

The addition of a regularising term λ allows us to cast the optimisation as a standard quadratic programming problem, i.e.

$$\hat{P}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T(A + \mathbf{1}\lambda)\,\mathbf{w} - \boldsymbol{\alpha}^T B\mathbf{w}$$
$$\text{s.t } \mathbf{w} \geq 0, \quad \mathbf{1}^T\mathbf{w} = 1 \tag{3.19}$$

where $\lambda > 0$ is a regularisation constant, and

$$A_{ij} = \langle \mu[\hat{P}_i], \mu[\hat{P}_j] \rangle, \tag{3.20}$$

$$B_{ij} = \langle \mu[\hat{P}_i], k(x_j, \cdot) \rangle. \tag{3.21}$$

As A is symmetric and positive semi-definite by construction, this is a convex quadratic program. The matrices A and B depend only on the choice of kernel and the choice of mixture components, not the distributions themselves. Therefore they are only computed once before inference begins. To recover an estimate of the posterior in terms of the mixture distribution, we need only perform this optimisation. Many fast algorithms for solving convex quadratic programs exist [84, 82, 38], though scaling beyond thousands of mixture components poses a challenge.

The choice of mixture components will determine the convergence properties of our algorithm. However, as long as we can bound the distance between the embedded KBR posterior and our pre-image embedding, our estimate will be bounded with respect to functions that are reasonably smooth in H_X(i.e. in the unit ball). This proof follows the technique used in Lemma 1 in [108].

Proposition 3.5.1. *Let $P(X|Y = y)$ be the posterior distribution after instantiation with Bayes' rule using a joint probability $P(X, Y)$ and a prior $P(X)$. Let $\mu[P_k] = \mathbf{k}_X[\cdot]^T \boldsymbol{\alpha}$ be the posterior embedding from KBR with training samples $\{(x_i, y_i)\}_{i=1}^N$ from $P(X, Y)$ and weighted prior samples $\{u_i, \gamma_i\}_{i=1}^M$ from $P(X)$. In the computation of the KBR posterior, let the regularisation terms $\varepsilon = N^{-2a/3}$ and $\delta = N^{-8a/27}$ for some $a > 0$. Let $\hat{P}$ be fixed mixture estimate of the posterior from equation 3.19, and let Δ be the RKHS distance between the embedding of $\hat{P}$ and the KBR posterior, $\Delta = \|\mu[P_k] - \mu[\hat{P}]\|_H$.*

Assume the convergence conditions for KBR hold from theorem 5 in [34]; specifically, assume that the estimate $\mu[\pi]$ of the prior converges to the true prior such $\|\mu[\pi] - \mu[P_\pi(X)]\| = O(N^{-a})$ for $0 \leq a \leq 0.5$. Also assume that P_π / P is in the range of $\sqrt{C_{XX}}$, and that $E_P[f(X)|Y = y]$ is in the range of C_{YY}^π.

Then for a function f in the unit ball of the RKHS H_X defined by the points $\{x_i\}$,

$$\sup_f |E_P[f(x)] - E_{\hat{P}}[f(x)]| = O(N^{-8a/27}) + \Delta. \tag{3.22}$$

Proof. By expanding this modulus in terms of the expectation of P_k, and from the triangle inequality,

$$\sup_f |\mathrm{E}_P[f(x)] - \mathrm{E}_{\hat{P}}[f(x)]| \leq \sup_f |\mathrm{E}_P[f(x)] - \mathrm{E}_{P_k}[f(x)]| + \sup_f |\mathrm{E}_{P_k}[f(x)] - \mathrm{E}_{\hat{P}}[f(x)]|. \tag{3.23}$$

From the KBR consistency proof, we know that under the stated assumptions, $|\mathrm{E}_P[f(x)] - \mathrm{E}_{P_k}[f(x)]| = O(N^{-8a/27})$. Now, using the reproducing property to expand the second term we have $\mathrm{E}_{P_k}[f(x)] = \langle f, \mu[P_k] \rangle$ and $\mathrm{E}_{\hat{P}}[f(x)] = \langle f, \mu[\hat{P}] \rangle$, and so

$$\sup_f |\mathrm{E}_{P_k}[f(x)] - \mathrm{E}_{\hat{P}}[f(x)]| = \sup_f |\langle f, \mu[P_k] \rangle - \langle f, \mu[\hat{P}] \rangle| \tag{3.24}$$

$$= \sup_f |\langle f, \mu[P_k] - \mu[\hat{P}] \rangle| \tag{3.25}$$

$$\tag{3.26}$$

Because f is in the unit ball in H, $\sup_f |\langle f, \mu[P] - \mu[\hat{P}] \rangle| = \|\mu[P] - \mu[\hat{P}]\|_H$ which implies $\sup_f |\langle f, \mu[P] - \mu[\hat{P}] \rangle| = \Delta$. Substituting ε into equation 3.23 completes the proof. ■

Ideally, we would like $f(x) = \delta(x - t)$ in this proof to demonstrate point-wise convergence to the true posterior. However, Dirac delta functions are not contained in H_X with finite samples. Nonetheless, we can bound the H-norm distance between the true Dirac embedding and the empirical estimate by $O(N^{-0.5})$ for vanishing kernel bandwidth (see theorem 2 in [108]). This ensures point-wise convergence of the PDF itself.

3.5.3. Mixture of Kernels Posterior Estimates

The choice of mixture components has, up to this point, been arbitrary. However, using the kernel itself as the mixture component simplifies the construction of the matrices A and B, and is intuitively reasonable barring any additional information. In this case, the posterior component $\hat{P}_i = ck(\mu_i, \cdot)$, where c is a normalising constant such that $c = (\int k(\mu_i, X)\, \mathrm{d}X)^{-1}$, and μ_i is the mean location of the i-th mixture component. The

matrices A and B then simplify to

$$A_{ij} = c^2 \iint k(\mu_i, s)k(\mu_j, t)k(s, t)\,\mathrm{d}s\,\mathrm{d}t, \tag{3.27}$$

$$B_{ij} = c \int k(\mu_i, s)k(X_j, s)\,\mathrm{d}s. \tag{3.28}$$

For the Gaussian kernel, the posterior representation is a Gaussian mixture, and these expressions have analytic form. Then we can write the kernel as

$$k_x(x, x') = (2\pi)^{k/2}|\Sigma_x|^{1/2}\mathcal{N}(x; x', \Sigma_x), \tag{3.29}$$

and the mixture components as $\hat{P}_i(x) = \mathcal{N}(x; \mu_i, \Sigma_i)$, where Σ_x and Σ_i are covariance matrices. The terms of A and B are then

$$A_{ij} = (2\pi)^{k/2}|\Sigma_x|^{1/2}\mathcal{N}(\mu_i; \mu_j, \Sigma_x + \Sigma_i + \Sigma_j), \tag{3.30}$$

$$B_{ij} = (2\pi)^{k/2}|\Sigma_x|^{1/2}\mathcal{N}(X_j; \mu_i, \Sigma_x + \Sigma_i). \tag{3.31}$$

Special Case: Mixtures of Training-centred Kernels

Further simplification can be gained if we take the means of the mixture components as the training points x_i. Then $M = N$ and

$$A_{ij} = c^2 \iint k(x_i, s)k(x_j, t)k(s, t)\,\mathrm{d}s\,\mathrm{d}t, \tag{3.32}$$

$$B_{ij} = c \int k(x_i, s)k(x_j, s)\,\mathrm{d}s, \tag{3.33}$$

where $c = (\int k(\mu_i, X)\,\mathrm{d}X)^{-1}$. For a Gaussian kernel and assuming all mixture components have standard deviation Σ,

$$A_{ij} = (2\pi)^{k/2}|\Sigma|^{1/2}\mathcal{N}(x_i; x_j, 3\Sigma), \tag{3.34}$$

$$B_{ij} = (2\pi)^{k/2}|\Sigma|^{1/2}\mathcal{N}(x_i; x_j, 2\Sigma). \tag{3.35}$$

Other Choices of Mixture Components

In our experiments in this chapter we fixed the location and parameters of the mixture components *a-priori* to reduce the number of free parameters in the algorithm. We utilised the kernel itself as a mixture component, and centring the mixtures at the training points. In this case, the form of the embedding and the posterior are identical, except that the posterior has positive normalised weights that ensure it is a valid probability distribution.

Using fixed mixture components is by no means the only strategy that could be employed with this algorithm. In chapter five, we experiment with learning the pre-image mixture components from the training data in the context of scaling to large datasets.

Other possible strategies could include uniform grids, or multi-scale methods in which successively finer mixtures are used to approximate regions of greater probability. These types of pre-images are the subject of future work. As with any mixture model, high-dimensional problems will require many components to represent a complex distribution. Similarly, the MKBR algorithm will perform poorly in regions away from training data (like any non-parametric method). However, it is worth emphasising that the choice of mixture components in no way affects the underlying inference process, and merely corresponds to different approximations of the embedded posterior.

3.5.4. Normed-Weights Posterior Estimation

Another, simpler approach to obtaining a density estimate from the posterior embedding of KBR is to still use training-point centred kernel mixtures as a candidate distribution, but simply clip and normalise the posterior weights directly to produce a valid PDF. This idea follows from the simplification of KBR used in [86]. There is currently no proof of convergence for this technique, but we demonstrate good empirical results. This method is also numerically much more efficient, requiring no optimisation step, and as such is used in chapter five to scale MKBR to very large datasets.

The expression for the joint embedding in KBR (equation 2.85) produces a weight vector $\boldsymbol{\beta}$ such that

$$\hat{C}_{XY} = \sum_{i=1}^{N} \beta_i k(X_i, \cdot) \otimes k(Y_i, \cdot), \tag{3.36}$$

where $\boldsymbol{\beta} = (G_{XX} + N\varepsilon I)^{-1} G_{XX}^{-1} G_{UX} \boldsymbol{\gamma}$ and $\boldsymbol{\gamma}$ is the vector of prior weights.

The subsequent posterior expression can be simplified if we approximate $\boldsymbol{\beta}$ with $\hat{\boldsymbol{\beta}}$:

$$\hat{\beta}_i = \frac{\max(\beta_i, 0)}{\sum_{j=1}^{N} \max(\beta_j, 0)}. \tag{3.37}$$

Now $\hat{\boldsymbol{\beta}}$ has the properties that $\hat{\boldsymbol{\beta}}^T \mathbf{1} = 1$ and $\hat{\beta}_i \geq 0 \forall i$.

Whilst this initial normalisation is not strictly necessary to produce a posterior estimate, the positive weights allow a simpler Tikhonov regularisation [122] to be used for calculating the KBR operator $R_{X|Y}$ from equation 2.97. We use $\mu[X|y=y] = \mathbf{k}_X^T[\cdot] \hat{R}_{X|Y} \mathbf{k}_Y[y]$, where

$$\hat{R}_{X|Y} = \left((D(\hat{\boldsymbol{\beta}}) G_{YY}) + N\delta I \right)^{-1} D(\hat{\boldsymbol{\beta}}), \tag{3.38}$$

and $D(\hat{\boldsymbol{\beta}}) = \mathrm{Diag}(\hat{\boldsymbol{\beta}})$. An analogous simplification occurs for the filtering prediction step.

Using the same clipping and normalisation again on the posterior weights, we can approximate the posterior embedding by $\mu[X|Y=y] = \mathbf{k}_X^T[\cdot] \hat{\boldsymbol{\alpha}}$, where

$$\hat{\boldsymbol{\alpha}} = \hat{R}_{X|Y} \mathbf{k}_Y[y], \tag{3.39}$$

$$\hat{\alpha}_i = \frac{\max(\alpha_i, 0)}{\sum_{j=1}^{N} \max(\alpha_j, 0)}. \tag{3.40}$$

For a stationary kernel, the embedding $c \mathbf{k}_X[\cdot]^T \hat{\boldsymbol{\alpha}}$ is also a valid probability distribution on $\mathcal{X}$, where $c = \left[\int k(0, x) \, \mathrm{d}x \right]^{-1}$. Rather than find the minimum of the quadratic program in equation 3.19, we can use it directly as a point in the viable set of the optimisation. With a suitable training scheme, this approximation has shown good empirical results in experiments.

We have now presented two methods for recovering posterior distribution estimates from embedded posteriors. We now give an algorithmic overview of how to compute these estimates in tandem with the KBR algorithm.

3.5.5. The Multi-Modal Kernel Bayes' Rule Algorithm

In this section we now give an algorithmic description of how these posterior recovery methods are used in combination with the KBR for performing both regression and filtering. Assuming we are using kernels as mixture components (as described in sections 3.5.3 and 3.5.4), we denote the combination of KBR with kernel mixture estimates as multi-modal kernel Bayes' rule (MKBR).

In the regression case, the goal is to estimate the conditional posterior $P(X|Y = y^*)$ of some random variable X, given an observation y^*. Rather than consider a single observation, we often want to evaluate a set of query points $\{y_i^*\}_{i=1}^T$. The corresponding output of the algorithm is a set of vectors $\{\mathbf{w}_i^*\}_{i=1}^T$ that define the posterior distribution at each query point y_i^*. For the filtering case, the conditional posterior is $P(X_t|Y{=}y_{1:t}^*)$.

The relationship between X and Y is learned from a set of training points $\{x_i, y_i\}_{i=1}^N$, in the regression case drawn from the joint $P(X, Y)$, and in the filtering case from the joint process $P(X_t, Y_t)$, i.e. $\{x_t, y_t\}_{t=1}^N$. The prior for X is estimated using samples $\{u_i\}_{i=1}^M$ from the prior $P(X)$ (or $P(X_1)$ in the filtering case) with associated weights $\{\gamma_i\}_{i=1}^M$.

The algorithm requires kernel functions k_X and k_Y to be given, which specify the particular RKHS in which the probability distributions will be embedded. We assume that the posterior mixture components are given by the kernels themselves centred at the training points, but if this assumption were relaxed then the components would also need to be specified.

In the regression case, there are a number of terms that can be pre-computed for all query points. These are the pre-image matrices A and B from equation 3.20, the joint embedding coefficients $\boldsymbol{\beta}$ in equation 2.85, and the $R_{X|Y}$ operator used to compute the posterior weights, in equation 2.97 or 3.38. Once these have been calculated, the algorithm loops through every query observation y_i^*, computes the embedding $\mathbf{k}_Y[y]_i = \sum_j^N k_Y(y_j, y_i^*)$, then calculates the posterior embedding $\boldsymbol{\alpha}_i$ from equation 2.96. Finally, using either equation 3.19 or 3.39, the posterior mixture weights are calculated. Algorithm 1 gives an overview of this procedure in pseudo-code.

The filtering case is very similar. The important point to note here is that recursive estimation of the embedding can occur without the need to compute the pre-image. The filter continues to iterate, with a pre-image calculation only performed when an estimate

Algorithm 1: The MKBR regressor described in pseudo-code.

Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$,
weighted prior samples $\{j, \gamma_j\}_{j=1}^{M}$,
kernels and parameters (k_x, θ_x), (k_y, θ_y) matrix regularisers ε, δ,
pre-image regulariser λ,
mixture components $\{\mu_k, \sigma_k\}_{i=1}^{Q}$ observation queries $\{y_k^*\}_{t=1}^{T}$
Output: posterior mixture weights $\{\mathbf{w}_t^*\}_{t=1}^{T}$

Calculate pre-image matrices A, B (Eq. 3.20);
Calculate joint $\boldsymbol{\beta}$ (Eq. 2.85) ;
Calculate $R_{X|Y}$ (Eq. 2.97 or 3.38);
for $i \leftarrow$ 1**to** T **do**
    Embed observation $\mathbf{k}_Y[y]_i = \sum_j^N k_y(Y_j, y_i)$;
    Calculate posterior $\boldsymbol{\alpha}_i$ (Eq.2.96);
    Find mixture weights $\mathbf{w}_i$ (Eq. 3.19 or 3.39);
end

of the distribution is required external to the algorithm.

The initial embedding is computed through the technique described in equation 3.12, and the pre-image matrices A and B pre-computed from equation 3.20. Because the estimation is recursive, we cannot pre-compute $R_{X|Y}$ as this operator relies on the prior, which is the posterior embedding from the previous time-step.

At each time-step, the predictive distribution embedding is calculated with equation 3.9. If there is no observation, this embedding is updated by embedding the observation, then computing $R_{X|Y}$ and the posterior, by equation 3.11. The posterior for this time-step is then set as the prior for the next time-step. Should an estimate of the pre-image be required, it is computed with equations 3.19 or 3.39. Note that in the filtering case it is also possible to use either method of pre-image recovery. Algorithm 2 gives a pseudo-code outline of the MKBR filter.

3.5.6. Computational Complexity

For N training points, T query points, the computational complexity of the regression algorithm is $O(TN^3)$. If the pre-image optimisation is performed as per equation 3.19, then this is the naïve complexity of solving the convex quadratic program. If the normed-

Algorithm 2: The MKBR filter described in pseudo-code.

Input: training set $\{(x_i, y_i)\}_{i=1}^{N}$,
weighted prior samples $\{\beta_j, u_j\}_{j=1}^{M}$,
kernels (k_x, θ_x), (k_y, θ_y) pre-image regulariser λ,
mixture components $\{\mu_k, \sigma_k\}_{i=1}^{Q}$ initial observation y_0
Output: posterior mixture weights $\mathbf{w}_{t+1}$

Calculate pre-image matrices A, B (Eq. 3.20);
Calculate initial embedding $\boldsymbol{\alpha}^0$ (Eq. 3.12);
while running do
  Calculate prediction $\boldsymbol{\alpha}_{t+1}^{P}$ (Eq. 3.9) ;
  if new observation y_{t+1} **then**
    Calculate $\Lambda = \text{diag}(\boldsymbol{\alpha}_{t+1}^{P})$;
    Embed observation $\mathbf{k}_Y(y)_i = k_y(Y_i, y_{t+1})$;
    Calculate posterior $\boldsymbol{\alpha}_{t+1}$ (Eq. 3.11);
  else
    Use prediction $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\beta}_{t+1}$;
  end
  if estimate then
    Find mixture weights $\mathbf{w}_{t+1}$ (Eq. 3.19 or 3.39);
  end
end

weights method is used as per equation 3.39, then the most expensive step is the matrix multiplication in equation 2.96.

The filtering case is also $O(TN^3)$, but there is an additional term with that complexity, which is computing $R_{X|Y}$ as per equations 2.97 or 3.38. Note that in the filtering case, both the input and output of the KBR are embeddings, meaning that the filter can run independently of when we choose to recover estimates. We could, for instance, only recover an estimate when a user queries, or when a particular condition is met. This would have no effect on the embeddings computed in the filter.

3.6. Parameter Learning

As seen in the previous section, there are a number of unknown parameters in the MKBR algorithm. There are two regularisers in the KBR, plus one in the quadratic program for recovering posterior estimates. Any parameters of the kernels used must also be determined. Finally, there are the parameters of the posterior mixture components; their positions and length-scales in each output dimension.

A common approach to learning unknown parameters in an algorithm is to define a cost function that characterises the quality of the algorithm's output, then use the unknown parameters as variables in an optimisation trying to minimise this cost function. The original KBR algorithm took this approach with a cost function defined as the mean-squared error between a maximum *a-posteriori* estimate of the posterior (the mode), and a training point. However, point statistics such as these poorly represent the structure of a probability distribution.

A more rigorous cost function would attempt to characterise the difference between the estimated distribution and the true posterior. Measures such as the Kullback-Leibler divergence or mutual information would be appropriate [78], but the requirement to know the true conditional posterior for the training data is unrealistic for most real problems.

Therefore, a compromise measure is needed that properly accounts for the shape and multi-modality of the posterior estimate, but that can be evaluated with access only to $\{x_i, y_i\}$ samples from the joint $P(X, Y)$. One choice that has been successful in the Gaussian process literature is the negative log-probability (NLP) [98]. This cost function measures

the log-probability of drawing a 'true' value x_i from the posterior distribution estimated by conditioning on y_i. Using the log of this quantity make evaluating it simpler with limited floating point precision, and taking the negative ensures that cost decreases with an increasing quality of the solution.

To define this cost function explicitly, let $\{(x_i, y_i)\}_{i=1}^N$ be samples from the joint $P(X, Y)$, and let $\hat{P}(X|Y=y_i; \theta)$ be the posterior estimate produced by the MKBR algorithm when conditioned on a training observation y_i, with unknown parameter vector $\theta \in \mathbb{R}^M$.

Then the NLP cost function is

$$C(\theta) = -\sum_{i=1}^{N} \log \hat{P}(x_i|Y=y_i; \theta). \tag{3.41}$$

Here θ is a vector composed of the M unknown parameters in the algorithm: the kernel widths σ_x and σ_y (section 2.9.1), the matrix regularisers ε and δ (section 3.4.1), and the pre-image regulariser λ (section 3.5.2);

$$\theta = (\boldsymbol{\sigma}_x, \boldsymbol{\sigma}_y, \varepsilon, \delta, \lambda). \tag{3.42}$$

In practice, the training function in equation 3.41 is prone to over-fitting. We therefore use k-fold cross-validation to evaluate it on hold-out sets of the training points. We divide the training data into k (approximately) evenly sized groups, then evaluate the cost function for each fold, using the other $k-1$ folds as the training data for that evaluation. If $I = 1 \ldots N$ and I_k is the set of indices of the kth fold in the training set $\{x_i, y_i\}_{i=1}^N$ then the resultant cost function is

$$C_{CV}(\theta) = \frac{-1}{N} \sum_{j=1}^{k} \sum_{i \in I_j} \log \hat{P}_{I \setminus I_j}(x_i|Y=y_i; \theta), \tag{3.43}$$

where $\hat{P}_{I \setminus I_j}$ is the distribution estimate trained on the joint training samples with indices in $I \setminus I_j$. Taking the mean over the testing points allows for comparison between different sized training sets. In other words, the cross validation function chooses a θ, then uses algorithm 1 to make posterior predictions which are evaluated with the NLP cost function on the hold-out data. The computational complexity of the training scheme is $O(N^4)$ per

iteration, due to the need to evaluate the hold-out sets for each fold.

The k-fold cross validated NLP described in this section is the cost function we use for all subsequent work in this thesis unless otherwise noted.

3.6.1. Implementation

Practically speaking, parameter learning is the biggest challenge to creating an efficient implementation of KBR, and so also of MKBR. Both the KBR algorithm and our extensions are prone to numerical instability from the Gram matrix inversions, and the parameter space generally contains local minima requiring robust global search strategies. Additionally, the computational cost associated with training can be high, as both the original cost function in [34] and the one developed above, partially due to the computation of the posterior weights, but also the optimisation required to recover the posterior estimate (or MAP estimate in the case of [34]. The use of cross-validation further exacerbates the issue.

Despite these challenges, a numerically efficient implementation and a good-quality global optimiser can demonstrate fast and reliable training for filtering and regression problems. We employ the global search algorithm know as multi-level single linkage (MLSL) [59] for the M-dimensional optimisation of θ, which seeds many local convex optimisations from random starting points in the search space. The implementation also utilises low-discrepancy sequences rather than random numbers, which improve the convergence rate by increasing the expected spacing between random draws [71]. For the local optimiser, we use Powell's canonical implementation of constrained optimisation by linear approximations (COBYLA) [93]. This is a derivative-free algorithm that constructs successive linear approximations to the cost functions.

Within the pre-image method, the convex quadratic program solver implementation utilises the method of moving asymptotes (MMA), globally converging algorithm utilising analytic derivative information [115]. All the implementations of these optimisers are part of the 'NLOpt' software package [54].

Numerical Stability

A significant problem in implementing an efficient training scheme is that many combinations of kernel width parameters and regulariser for the Gram matrix inversion will

result in the required linear solve not converging (for instance, equations 2.97 and 2.85). This yields a numerical error in the implementation, effectively ruling out gradient-based search algorithms, as the search must be re-started from a different point.

We take a number of steps to alleviate this issue. The first is to never explicitly compute a matrix inverse, but instead to perform Cholesky decompositions and solve the equivalent linear system. We use a Cholesky decomposition routine which to solve the linear system $\mathbf{x} = A^{-1}b$ decomposes A such that $A = LL^T$, then computes $L\mathbf{y} = b$ and $L^*\mathbf{x} = \mathbf{y}$. This is much more stable numerically than explicitly computing the inverse matrix A^{-1}.

We also use a variant of the 'jitChol' algorithm used in the venerable GPML Gaussian process MATLAB implementation [98] to increase stability. JitChol is a function that attempts to solve the linear system associated with the requestion matrix inversion. If the solver returns an error, then a jitter or regulariser is added to the lead diagonal of the matrix, and the solve re-attempted. The jitter is doubled until the solver returns a valid solution. The aim here is to remove any small negative eigenvalues from the matrix that are preventing the solver from successfully completing a Cholesky decomposition.

Our approach extends jitChol; rather than using the successful return of the linear solver, we compute the error bounds of the inversion explicitly by considering $||Ax - b||$. If a fixed accuracy bound is not satisfied then jitter is added. We also implement a smart caching system for remembering the size of previous jitter additions, to minimise the number of times the inversion has to be re-computed to meet the error criterion.

3.7. Experiments

To demonstrate the effectiveness of the MKBR filter and regressor, we performed several experiments modelling motion in robotics applications designed to require multi-modal, non-linear estimation.

The first experiment was a multi-modal tracking simulation of an entity able to take multiple routes around a fixed track decided by coin tosses at the junctions, with infrequent observations. The resultant distribution in space for the entity must be multi-modal until the track it took at the most recent junction was observed. The performance of our filter was compared with variations of a standard particle filter given the true underlying mod-

els (and even the coin tosses) [24], as well as a GP-Bayes particle filter that uses Gaussian processes to model the underlying prediction and observation distributions [65].

The next experiment was another filtering challenge involving real data taken from an inertial measurement unit (IMU) attached to a slot car as it looped around a fixed track. The challenge was to predict the rate of progress of the car along the track given only raw IMU data, effectively learning the car's motion model and implicitly the geometry of the track. Our filter was again compared against the GP-Bayes particle filter.

Finally, the third experiment was a regression problem addressing estimation of a direction field associated with track of pedestrians moving in an indoor area. A probabilistic map of allowable velocities was created, and used by a set of simulated robots to perform path planning. Our algorithm was compared against the published implementation [88] which uses a GP to model the direction field [98].

We now give a brief overview of the algorithms against which we compared our techniques, before detailing the experimental setup and results.

3.7.1. Comparison Algorithms

For the experiments, we compared MKBR against standard algorithms used in robotics for filtering in regression:

Gaussian Process Regression

Gaussian processes regression (GPR) is a generalisation of multi-variate Gaussian regression, to potentially infinite-dimensional function spaces [98]. Unfortunately, whilst the kernel literature including the original KBR paper [34] denote a true state as x and a (possibly noisy or indirect) observation with y, the GP community do the opposite. At risk of confusing those more familiar with the latter, we will use notation consistent with the rest of the work in this thesis, which follows the x as state, y as query convention.

GPR requires a set of training points $\{x_i, y_i\}_{i=1}^N$ and a covariance function $k(\cdot,\cdot)$ in the form of a positive-definite kernel. Given a set of query points $\{y_i^*\}_{i=1}^M$, the GPR algorithm returns an estimate of $P(X|Y = y_i^*)$, assuming that the underlying relationship between X and Y is $X = f(Y) + \varepsilon$, where f is an arbitrary function, and ε is Gaussian-distribution noise with mean 0 and variance σ. GPR assumes the query points and the training data

are actually a single point drawn from an $N + M$-dimensional Gaussian distribution, with mean 0 and covariance defined by

$$\begin{bmatrix} x \\ x^* \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} G & G_*^T \\ G_* & G_{**} \end{bmatrix}\right) \tag{3.44}$$

where G is the covariance (Gram) matrix $G_{ij} = k(x_i, x_j)$, $(G_*)ij = k(x_i, X_j^*)$ and $(G_{**})_{ij} = k(x_i^*, x_j^*)$. Conditioning on the training data yields a Gaussian distribution at the query points with mean

$$\mathrm{E}[x*] = G_* \left[G + \sigma I\right]^{-1} \mathbf{y} \tag{3.45}$$

and variance

$$\mathrm{var}[x^*] = G_{**} - G_* \left[G + \sigma I\right]^{-1} K_*^T. \tag{3.46}$$

Gaussian process regression is able to represent complex, non-linear functions, but cannot represent multi-modal distributions, as the predicted distribution at every query point is a Gaussian. It is also non-trivial to extend GPs to the case where X is more than one-dimensional, although this is an area of current research [13, 4].

Computationally, GPR is dominated by the $O(N^3)$ matrix inversion with the number of training points. Evaluating a query point is a matrix-vector multiplication, which is $O(N^2)$ in general.

Particle Filter

Particle filters are a sequential Monte-Carlo approach to solving the Bayesian filtering problem by approximating distributions with a set of samples or particles, and propagating these through the prediction and observation models [12, 24]. These models must be known *a-priori*.

Given a state transition model $P(X_{t+1}|X_t)$ and observation model in $P(Y_{t+1}|X_{t+1})$, a particle filter produces a state X at time $t + 1$ conditioning on a set of observations $\{y_i\}_{i=1}^t$ as per equation 3.1.

One ubiquitous and successful variant of particle filtering is the sequential importance re-sampling (SIR) filter [24]. Given a set of samples $\{(x_i)\}^t$ which approximate $P(X_t|Y_{1:t} = y_{1:t})$ the SIR particle filter computes a set of updated samples $\{(x_i)\}^{t+1}$ approximating the distribution $P(X_{t+1}|Y_{1:t+1} = y_{1:t+1})$.

First, a new set of particles are sampled from the proposal distribution, usually the prior prediction probability, by

$$x_i^{(t+1)} \sim P(X_{t+1}|X_t = x_i^t) \tag{3.47}$$

given y_{t+1}, the importance weights of these samples (up to a normalising constant) are calculated through the observation model:

$$w_i^{(t)} = w_i^{(t-1)} P(Y_{t+1} = y_{t+1}|X_{t+1}). \tag{3.48}$$

These weights are then normalised such that $\sum_i w_i = 1$. The weighted samples now represent an estimate of the distribution $P(X_{t+1}|Y_{1:t+1})$. To ensure that particles are located in regions of high probability mass, we can re-sample the particles to obtain an un-weighted estimate. Simply drawing from the particles with probability given by w_i and with replacement yields a new particle population.

Common techniques used to estimate the distribution represented by the particles include Parzen windows, and histograms based on spatial partitioning. Computationally, the particle filter is a linear algorithm in the number of particles, but the particle count required for a given accuracy scales exponentially with dimensionality.

GP-Bayes Particle Filter

The GP-BayesFilter is a multi-modal, non-parametric filter by using banks of Gaussian process regressors to learn the prediction and observation models for a SIR particle filter [65]. Given a set of samples $\{x_t, y_t\}_{t=1}^N$ of state and observation pairs, the GP-BayesFilter approximates the state transition and observation models with

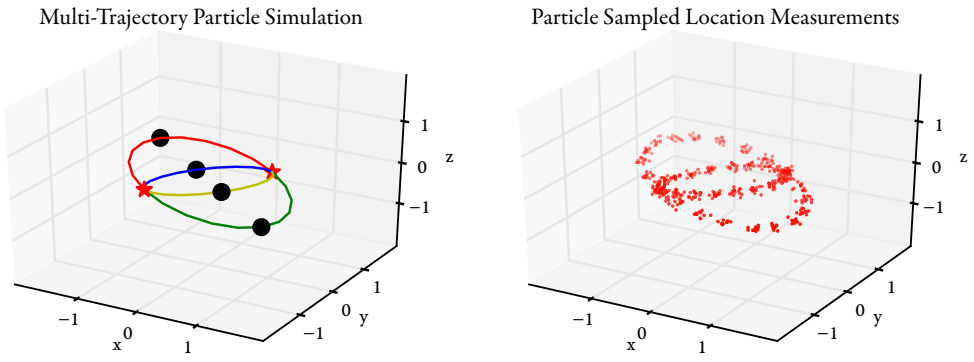


Figure 3.4.: (Left) The eye-shaped multi-trajectory particle simulation used to test multi-modal estimation. Red stars represent the points where the particle decides between the blue and red trajectories, and again between the yellow and green trajectories. Observations are taken only at points represented by black dots. (Right) An example of training data drawn from the simulation.

$$\hat{P}(X_{t+1}|X_{1:t}) = \mathcal{N}\left(X_{t+1}; \mu_p, \Sigma_p\right) \tag{3.49}$$

$$\hat{P}(Y_{t+1}|X_{1:t+1}) = \mathcal{N}\left(Y_{t+1}; \mu_o, \Sigma_o\right), \tag{3.50}$$

where the means and covariances are given by Gaussian process regressors, trained on (x_{t+1}, x_t) pairs for the prediction model and (x_t, y_t) pairs for the observation update.

If either X or Y is multi-dimensional, then a separate GP is required for each dimension. Computationally, the GP-BayesFilter particle filter is dominated by $(P+Q)$ Gaussian process evaluations for every time-step (where $X \in \mathbb{R}^P$ and $Y \in \mathbb{R}^Q$), which are each $O(N^3)$ in the number of training points if attempted naïvely, though it is possible to achieve $O(N^2)$ through sequential updating.

3.7.2. Experiment 1: Multi-modal Tracking Simulation

Experimental Setup

This experiment simulated a particle moving around two concentric loops. At the intersections of these two loops, the particle randomly chooses to follow one loop or the other.

However, observations of the particle only occur when it is half-way round the loop, which means that a filter predicting the location must consider both possible trajectories. This problem is often encountered in the robotics localisation domain, in which predictions of a target moving around an obstacle must consider the different ways in which that obstacle could be avoided. This simulation is illustrated in figure 3.4.

The equations of motion of the particle in 2-D were

$$x = \Big(\cos(2\pi/20t), (0.5 + 1.5\eta)\sin(2\pi/20t)\Big) + Z_t, \tag{3.51}$$

where η is a Boolean-valued random variable that was re-drawn at $t = 0, 10, 20, \ldots$ and Z_t is zero-mean Gaussian process noise with $\sigma_p = 0.05$. Observation noise is similarly distributed with $\sigma_o = 0.02$, and observations were taken every five time-steps.

We tested with problem dimensionality from 2-D to 10-D. For dimension $n > 2$, the plane of the loops was subject to n random sub-plane rotations using the formula $X' = RX$, where

$$R = \prod_{i=1}^{n} R_i(\phi_i), \tag{3.52}$$

$$(R_i)_{I_1 I_1} = \cos\phi_i \tag{3.53}$$

$$(R_i)_{I_2 I_2} = \cos\phi_i \tag{3.54}$$

$$(R_i)_{I_1 I_2} = -\sin\phi_i \tag{3.55}$$

$$(R_i)_{I_2 I_1} = \sin\phi_i. \tag{3.56}$$

The noise was computed in n dimensions after the rotations had occurred. The angle ϕ_i is drawn from $(0, 0.5\pi)$, I_1 and I_2 are random indices from 1 ... n, and the other entries of R_i are zero.

Comparison Algorithms

We compared our MKBR filter with a number of variations of particle filters, using the average log-probability (LP) of the test data under the predictive posterior [98]. As discussed in section 3.6, this measure properly accounts for complex posterior distributions

that may be multi-modal. The more probability mass an algorithm has assigned to the test point, the greater the performance. As the total probability mass for the predictive distribution is constant, this measure allows comparison between different algorithms—essentially measuring how likely each algorithm considered the test data to be.

Given test data $\{x_i^*, y_i^*\}_{i=1}^M$ the LL of a posterior estimate $P(x|Y=y_i^*)$ is

$$\mathrm{LL} = \frac{1}{M}\sum_{i=1}^{M} \log\Big[P(x_i^*|Y=y_i^*)\Big]. \tag{3.57}$$

The first algorithm, and comparable with ours, is the GP-BayesFilter particle filter, in which the process and noise models are estimated using banks of GPs trained on 600 samples from the trajectory. This algorithm is denoted GPPF in our results.

We also considered variations of the standard particle filter, actually giving the exact state transition and observation models from the simulation. The first, denoted PF, was not given the value of the random variable η that determines which path was being taken. This meant that the prediction model assigned particles to both possible paths. The second variation, denoted EPF, was given the value of η at every time-step. This meant that the EPF had the exact transition and observation models, and never even needed to make a multi-modal estimate.

All these particle filters were tested with 500, 1000, 5000 and 10,000 particles. Probability distributions were estimated at each time-step using a Gaussian Parzen window estimate. Two variations on bandwidth selection were tested, the normal approximations (Silverman's rule) [106], and an *a-posteriori* optimisation to maximise the probability, denoted by GPPF-O in our experiments. Though such an optimisation is hardly fair, it does place an upper bound on the possible accuracy of the particle filter methods.

We examined three variations of our filter. The KBR-N filter used the normed-weights method for approximating the posterior distribution. The KBR-NS method also used normed-weights, but pre-scaled all data to have mean zero and standard deviation one. The KBR-J used the full pre-image estimation, with kernel mixtures centred on the training points, and a joint optimisation of the kernel widths and pre-image regulariser. All variations utilised a spherically symmetric Gaussian kernel.

The MKBR filters were given the same 600 training points as the GPPF. The unknown

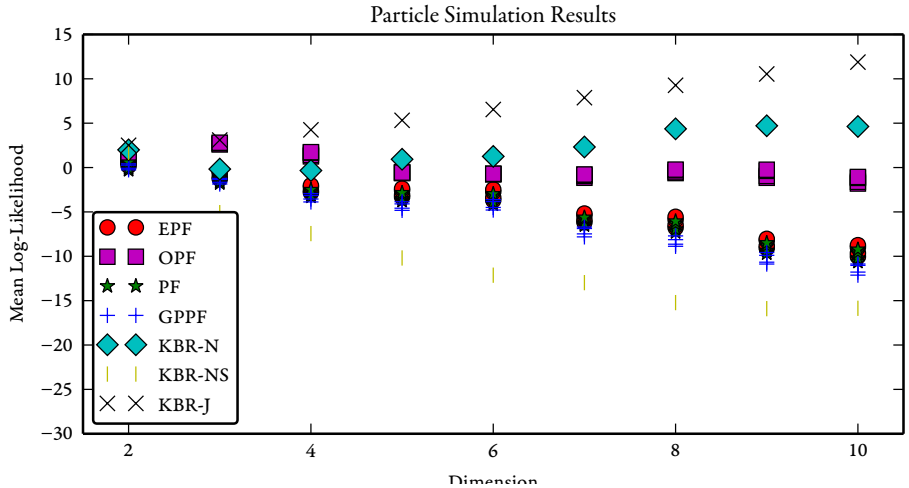


Figure 3.5.: Results for the multi-modal particle simulation. The sets of four points in each colour indicate results for 500,1000,5000 and 10,000 particles. In every case the accuracy of the result increased with particle count.

parameters were trained using the procedure outlined in section 3.6.1. The GPs in the GPPF were trained with the standard marginal likelihood method [98]. We used Python implementations of the Gaussian process [85] and the SIR particle filter[1].

Results

The results of this simulation are plotted in Figure 3.5, and tabulated in the appendix in table B.1. For all the particle filters, larger numbers of particles increased the accuracy of the posterior estimates as expected, and the dimensionality of the problem saw a corresponding decrease in performance. Unsurprisingly the filter not given access to the latent transition variable η (PF) performed worse than the EPF and OPF which did have access to this information. The OPF performed significantly better than the EPF, illustrating the difficulty of estimating the Parzen window bandwidth: The EPF estimated the bandwidth using Silverman's rule, whilst the OPF optimised it *a-posteriori*. This difference was enhanced in higher dimensions.

The GPPF performed the worst of all the particle filters. Given that it was forced to

[1]ProbRob version 28, http://launchpad.net/probrob

represent the motion and observation models using a uni-modal set of GPs, this result is to be expected. A GP-based solution to this problem would require explicitly modelling the latent variable, and the two separate paths.

The variations of our MKBR algorithm displayed significant differences in performance. The NS variation, which used normed weights and normalised the data to a mean of zero and unit standard deviation in each dimension, performed the worst of all algorithms tested. The likely explanation is that the re-scaling distorted the ring structure, making points at the same location in the ring, spread out over a whole dimension, now on the same scale as points on opposite sides of the ring, but in the plane. Therefore a tension was introduced in which the kernel width needed to be large in some dimensions and small in others. Our experiment used spherically symmetric kernels which meant this was impossible, and a compromise resulted in poor performance. Having more complex kernel functions that could vary in length scale per dimension would likely alleviate this problem.

The two other variations of the MKBR algorithm, the MKBR-N variant which used the normed weights pre-image method, and the MKBR-J variant which used the full quadratic program pre-image, displayed the highest performance. The -J variant was unbeaten in all dimensions, whilst the -N variant overtook the OPF particle filter from 5D onwards. The superior performance of these algorithms must come down to the underlying representation of the posterior distribution, as the MKBR methods were at a disadvantage in terms of accurate prediction and observation models. The uniformly weighted point-based method of representing distributions employed by particle filters are known to perform poorly with dimension [5].

The log-probability results for the MKBR -N and -J variants actually *increased* in performance with dimension. This result is more an artefact of the behaviour of the log-probability function in high dimensions, rather than a real increase in performance. The behaviour can be explained by recalling a property of N-dimensional Gaussians. For small variance, probability mass gets increasingly concentrated near the mean as dimensionality increases. Consider the N-D Gaussian PDF with spherically symmetric covariance $\Sigma = \sigma I$:

$$P(X) = (2\pi\sigma^2)^{-0.5D} \exp\left(-\frac{||x-\mu||^2}{2\sigma^2}\right). \tag{3.58}$$

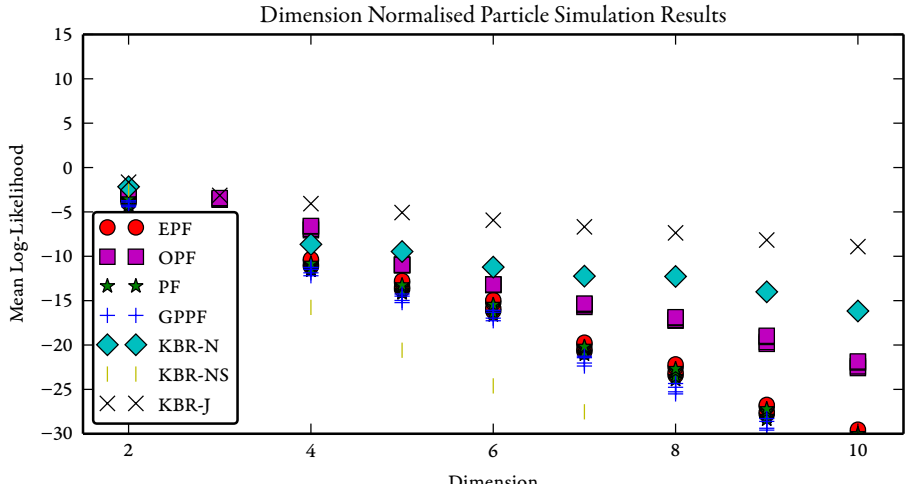


Figure 3.6.: Results for the multi-modal particle simulation with the $D \log 8$ factor subtracted to remove dimensionality effects as per section 3.7.2.

In our training, $\sigma \approx 0.05$ for all dimensions. Therefore the PDF for a mixture component was approximately

$$P(X) \approx 8^D \exp\left(-\frac{||x-\mu||^2}{2\sigma^2}\right). \tag{3.59}$$

As long as the accuracy of the simulation (as measured by the norm between the true point and the nearest mixture component) was increasing reasonably slowly, we would expect to see an increase in the probability with dimension from the scaling factor 8^D. The same results plotted with this factor subtracted can be seen in figure 3.6.

The -J variation of the MKBR with the full pre-image estimate found by joint optimisation performed the best of all the measures. This is likely due to the increased regularisation of the solution from the additional parameter in the convex optimisation (eq. 3.19). The normed weights solution tended to blur out the position of its estimate over a couple of time steps, an effect that was reduced by the pre-image regulariser, hence increasing the accuracy of the posterior estimate.

Analysing individual time steps indicates that the MKBR algorithms were able to learn the behaviour of the particle including the existence of the latent variable. The algorithm

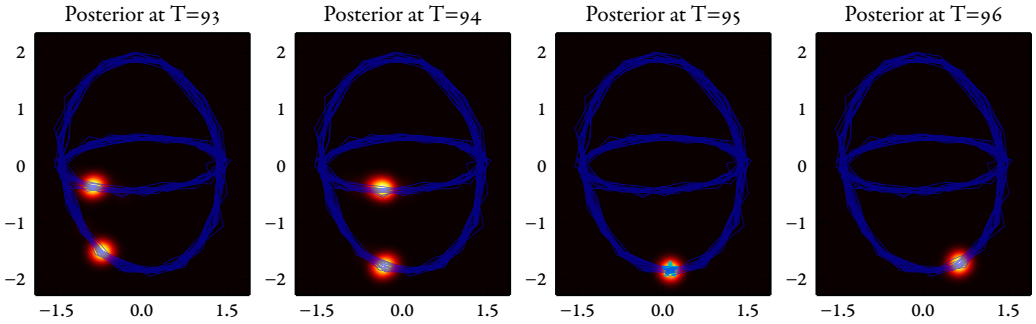


Figure 3.7.: The MKBR filter is able to learn the multi-modal path, which then collapses when an observation is made at t=95. Now that the measurement has been made the particle's trajectory is determined for the rest of the loop, so a uni-modal prediction is sufficient. The MKBR learns this behaviour directly from the data. The star at $T = 95$ indicates an observation made at that time-step.

knew that once the particle is observed in one track, it would stay in this track even though it was no longer being observed. This behaviour is illustrated in figure 3.7.

3.7.3. Experiment 2: Inertial Slot Car

Experimental Setup

The second experiment involved a miniature slot car moving around an 11-metre track with loops and banked curves. Inertial data was taken with a small IMU attached to the car. An overhead camera provided ground truth for the position of the car, which was interpreted as a scalar quantity equivalent to (un-normalised) proportion of the track complete. The derivative of this quantity is the norm of the car's velocity vector, or its velocity in the direction of motion. The goal of the experiment was to predict the track velocity of the car, using the six-dimensional IMU data as observations. The name of the dataset was "Slot Car Inertial Measurement", taken from [111].

The relationship between the IMU and the forward velocity of the car is complicated by the track; the car changes speed depending on the banking and slope of the track. Given

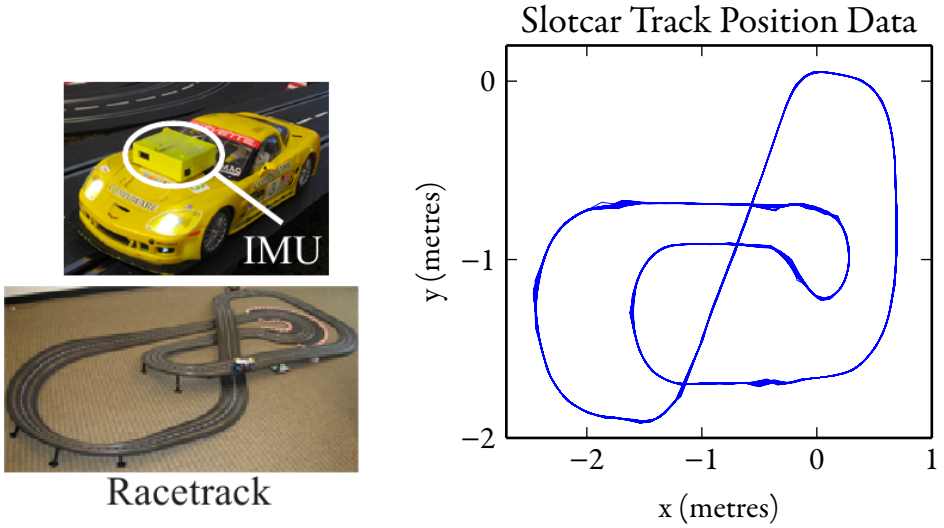


Figure 3.8.: The slot car experimental setup. A slot car was fitted with an IMU as it was driven around a fixed track.

the variability of these features in the track, the resulting likelihoods are non-trivial (time-varying) functions of the IMU variables. To compare performance of the algorithms we used the probability of the true state in the filter's posterior estimate which properly accounts for the generality of the distributions output by our filter.

Algorithms

We compared the performance of our algorithm with the GP-BayesFilter particle filter [65]. Both algorithms were estimating a full posterior over the car's position. Though there are many possible filters with which to compare, we are focussing on two filter properties: The ability to learn observation and transition models from data, and the ability to represent arbitrary posterior distributions. Standard filters such as the EKF are ruled out on both counts, and have previously been compared with the original KBR in [34], performing poorly in non-Gaussian problems such as this.

The filters were both given 600 data points for training corresponding to approximately five loops around the track. The GPPF used 500, 1000, 5000 and 10,000 particles. The ground truth for the experiment was given as a 1-D filtered track velocity from the over-

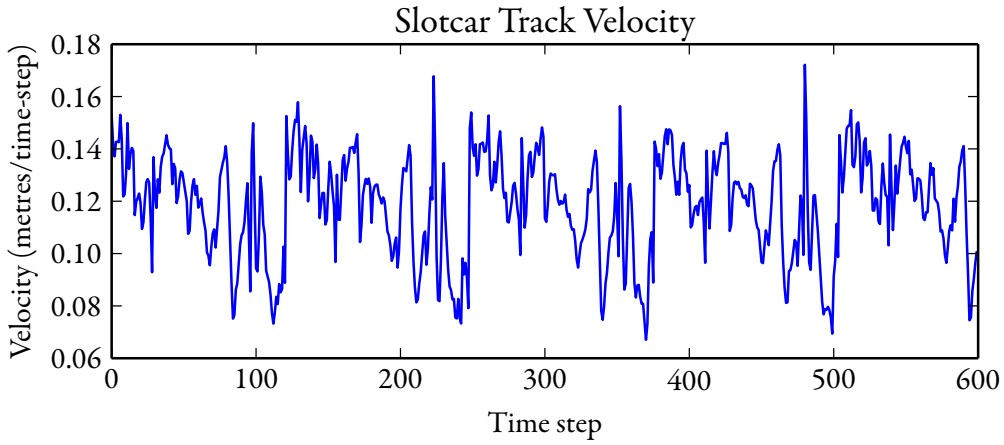


Figure 3.9.: The slot car training data, measured in metres per time step. This is the training data for the algorithms.

head camera. Measurements from the IMU were six-dimensional vectors of pitch, yaw, roll, and x, y, z accelerations. We tested on 600 data points. Posterior distributions for the GPPF were obtained from the particles via a Parzen estimator, with bandwidth set using two methods. The first, denoted GPPF, used the normal approximation, otherwise known as Silverman's rule of thumb. The second, denoted OPF, used an optimiser to determine the bandwith that maximised the posterior probability. The GP implementation used for both of these variations utilised the maximum marginal likelihood estimation method for learning GP hyperparameters from the training data, following [98].

Results

Table 3.1 lists the results. Both MKBR algorithms outperformed the GPPF in this measure. As expected, the accuracy of the GPPF algorithms increased with particle count, but it was still not able to compete with our methods, even with 10,000 particles. Though the GPPF was able to represent non-Gaussian posterior estimates, the underlying motion model relating the IMU variables was Gaussian. Furthermore, each dimension of the motion model was modelled by a separate GP, meaning there could be no covariance between different dimensions of the model. This contrasts with our method which can represent

the multi-dimensional, non-Gaussian prediction model as a high-dimensional RKHS embedding. Once again, the choice of Parzen window bandwidth had a significant effect on the performance of the GPPF models. The PF variation which used Silverman's rule performed significantly worse than the OPF which learned the bandwidth *a-posteriori*.

Figure 3.10 illustrates the posterior estimates of the different algorithms. The PF estimated the bandwidth too low, meaning the resultant distribution was over-confident. The optimised bandwidth was very large, owing to the low accuracy of the model representation. This again meant a low log-probability at the test locations. A single slice of the posterior at $T = 320$ emphasises this effect, shown in figure 3.11.

Unlike the previous experiment, in this instance the normed weights pre-image method (denoted by N) outperformed the full pre-image estimation algorithm (denoted by J). This suggests that posterior estimates generated by the normed weights method may have different characteristics from the full pre-image estimator that might be beneficial to certain types of problem. Understanding this intriguing result more fully is the subject of future work.

Another, more mundane explanation is that the parameter learning was not sufficient for the full pre-image algorithm. As each iteration of the cross-validation contains a convex quadratic program, it is a very time consuming process, and may not have run for long enough to find a good optimum for the algorithm parameters θ. For more details on the cross validation scheme, see section 3.6.

3.7.4. Experiment 3: Pedestrian Velocity Field

The third experiment involved generating a normalised velocity field for the purposes of indoor robot navigation [88]. Pedestrians moving around an indoor office space were tracked using a SICK laser. From this data, tracks of their position as a function of time were computed. At each point on the track, the direction of motion of the pedestrian can be estimated. Building up this velocity information for many pedestrians provides data about common paths through the area. These paths are useful for robots later navigating in the area, as it allows them to take advantage not only of a human's avoidance of obstacles which might be difficult to detect, but also because it gives the robot information about social boundaries such as office cubicles which might not normally be used as a thorough-

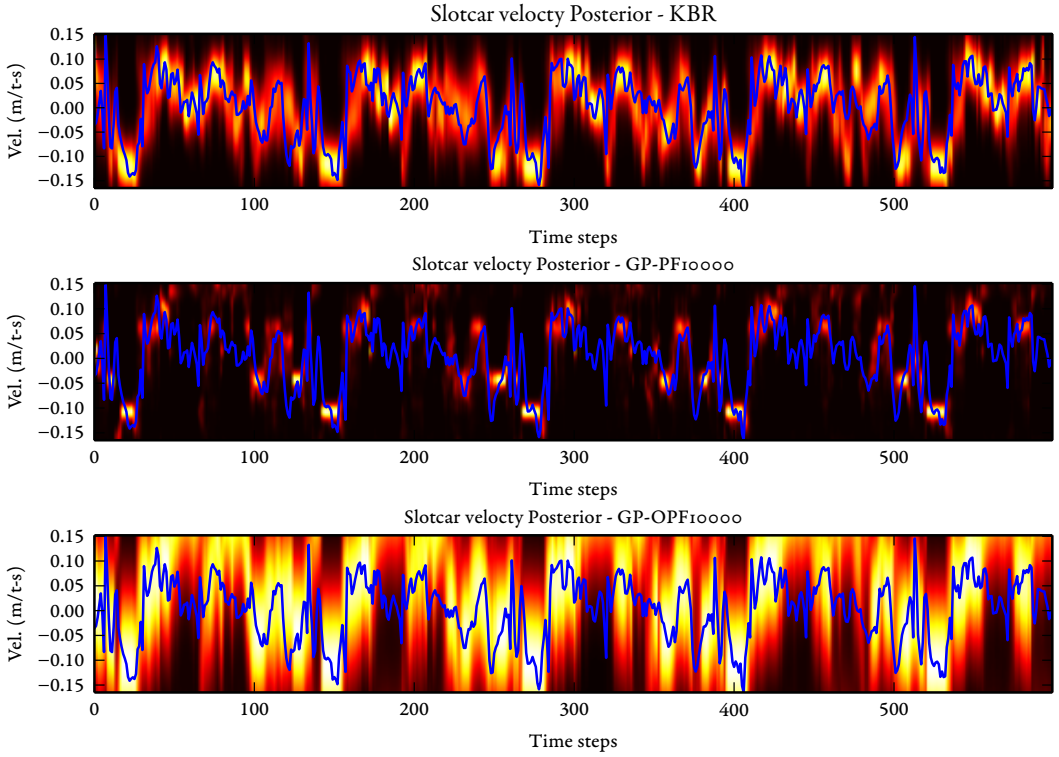


Figure 3.10.: Posterior distributions of the slot car data. (Top) MKBR posterior. (Middle) GPPF algorithm with 10,000 particles and Parzen bandwidth estimated by Silverman's rule. (Bottom) GPPF algorithm with 10,000 particles and bandwidth estimated *a-posteriori*.

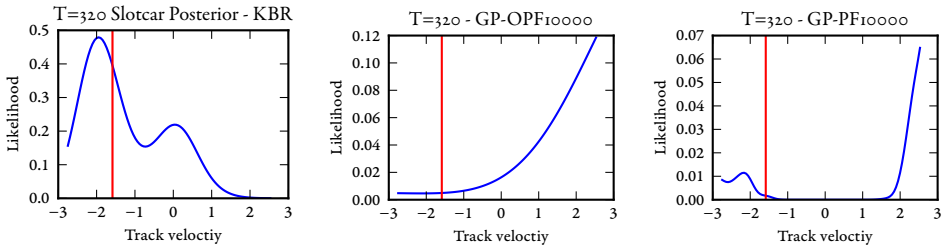


Figure 3.11.: Posterior distribution of the slot car data at $T = 320$. (Left) MKBR posterior. (Middle) The GPPF posterior with Parzen bandwidth optimised to maximise log-probability. The optimal bandwidth here is large, to compensate for the low accuracy of the filter relative to the ground truth. (Right) GPPF with Parzen bandwidth determined by the normal approximation. Here we see the bandwidth is probably too low. In each graph the true value is shown with a vertical line.

Table 3.1.: Results for the slot car dataset

Algorithm	Mean log-probability
GP-PF500	-6.279 ±10.032
GP-PF1000	-7.465 ±11.504
GP-PF5000	-8.716 ±16.665
GP-PF10000	-9.532 ±19.091
GP-OPF500	-2.167 ±0.792
GP-OPF1000	-2.238 ±0.744
GP-OPF5000	-2.142 ±0.812
GP-OPF10000	-2.146 ±0.806
MKBR-J	-1.346±1.109
MKBR-N	**-1.013 ±0.720**

fare. The data for this experiment were taken from the UTS RobotAssist project [62]. The inputs Y of the algorithm were taken as 2D positions in the room, and the underlying states X were the angles of the velocity field, represented as quaternions to avoid interpolation issues across the angle discontinuity.

We compared our MKBR algorithm to the original implementation in [88] which used a Gaussian process to model the direction field as a deviation from some prior model. In

Table 3.2.: The results of the pedestrian dataset.

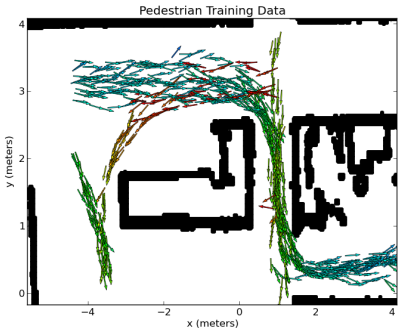
Algorithm	Mean log-probability
GP	2.052
MKBR	**2.903**

their experiment, the prior was a normalised velocity, which at every point, was aligned towards a single destination point. We have elected to make the destination an area rather than a point which corresponds to the yellow box in figure 3.13. As a result, the prior velocity field pointed straight down everywhere in the test area. For the GP, this involved setting the prior mean to $-\pi/2$, and for the MKBR, we created an embedded prior distribution from Gaussian samples centred around $-\pi/2$. 600 training points were given to both algorithms, which were randomly sampled from all available tracks which ended in the area of interest. The tracks were smoothed using a 11-point Hamming window before calculating velocity. To allow for meaningful interpolation, the angles were expressed in quaternions before being given to the algorithms. Note that this mean that two GPs were required, as in the original implementation. 4000 other points were sampled from the smoothed tracks for testing.

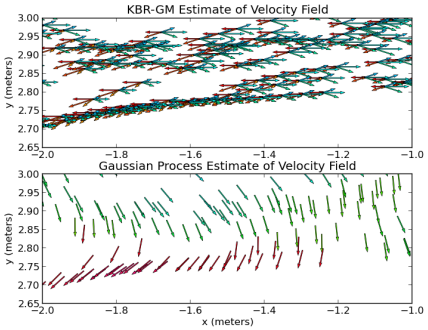
The GP hyper-parameters were trained using the maximum marginal likelihood, with a convex optimiser following [98]. The MKBR used 40 mixture components for the pre-image, evenly distributed over $(-\pi, \pi)$. The results of the experiment appear on table 3.2.

The MKBR algorithm outperforms the GP, primarily because of its ability to represent multi-modal distributions. The corridor at the top of the image in figure 3.12 had people walking in both directions to reach the same destination. As the GP is only able to learn a uni-modal posterior, the result is a weighted average of the two directions, which in this case points straight down. On the other hand, the MKBR is able to learn a bi-modal distribution which points both left and right. See figure 3.12 for close ups of this behaviour on the testing data.

To illustrate how this result would be useful to a real robot, we performed a simulation of a simple indoor robot using the posterior direction field learned by the algorithms to navigate from a starting position to the goal area. The robot first evaluated the posterior



(a) The training data derived from SICK laser scans of pedestrian pose



(b) (Top) mixture components are plotted as arrows. The distribution is bi-modal left and right. (Bottom) The GP must average the tracks going left and right, with erroneous result

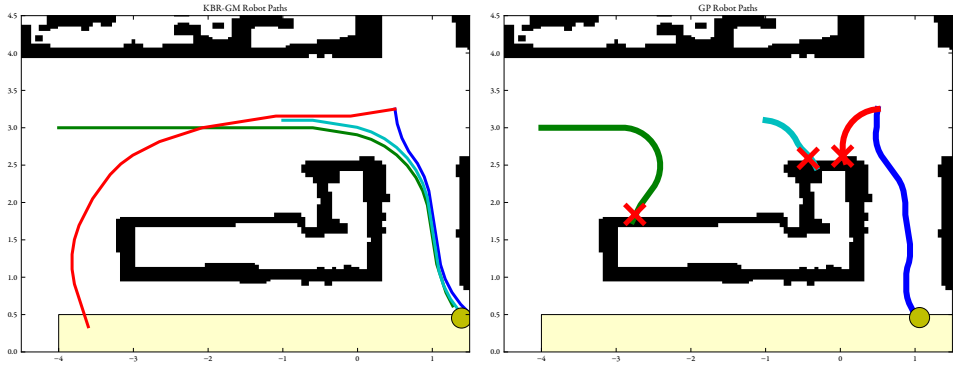
Figure 3.12.: Training data and posterior inset.

at its location. The robot then moved in the resulting direction for about 0.2 metres, and re-evaluated the posterior. In the case of a multi-modal posterior, the robot used the mode which was within π of its current orientation.

Figure 3.13 depicts these robots navigating from various starting locations and orientations. Notice that in the MKBR case, robots are able to move down the top corridor in opposite directions, whilst the robots using the GP are caught by the averaging of the two directions and as a result hit the wall.

3.8. Summary

In this chapter we have presented a new algorithm for Bayesian regression and filtering, MKBR. By building on kernel Bayes' rule, we have created an algorithm that can perform tractable Bayesian inference on non-Gaussian, multi-dimensional distributions, learning those distributions directly from sampled data. The output of the algorithm is a full posterior estimate in the form of a mixture distribution. Our algorithm works by solving the pre-image problem for the posterior embeddings generated by KBR. It embeds a mixture distribution with fixed components and variable weights, then determines the optimal



(a) Using the MKBR direction map the robots successfully navigate to the finishing area.

(b) The unimodality of the GP direction map causes some robots to collide with the kitchen bench.

Figure 3.13.: Robot navigation from direction maps

weights of the mixture to minimise the RKHS distance between the posterior embedding and the embedding of the mixture. These full posterior outputs required a new training scheme, and we proposed one based on minimising the negative log-probability of the training data, with k-fold cross validation to prevent over-fitting.

We demonstrated the effectiveness of our MKBR algorithm on a number of difficult motion estimation problems in robotics, comparing its performance to standard algorithms used in that domain. The MKBR learned latent variables in a motion model and was able to represent high-dimensional, multi-modal probability distributions. We also created a significant extension of a robotic path planning algorithm that used probabilistic velocity fields generated from pedestrian data [88]. This outperformed the original algorithm, demonstrating the ability to use training data containing conflicting direction vectors. This enabled robots to perform much more flexible path planning.

Chapter 4

Conditional Quantile Estimation from KBR Posterior Embeddings

Understanding risk is critical to making decisions based on uncertain information. Computing uncertainty through the use of probability, and then bounding that uncertainty with quantile estimation is one approach to this problem. Building from our work in chapter three, we now aim to extend our non-parametric, multi-modal techniques for regression to include cumulative and quantile estimation. With these additions, our work can be applied to problems with unknown prior and likelihood and produce not only probabilistic estimations, but also quantify the risk associated with decisions that rely on them.

4.1. Motivation

Probabilistic risk calculations are commonplace in any discipline making decisions under uncertainty. This includes environmental science, economics, geology, and engineering, which must all make potentially life-threatening choices about quantities that cannot be known precisely ahead of time. A model for climate change for instance, cannot predict with total accuracy the sea level rise resulting from any given global temperature change. It may not even be possible to be certain about reasonable bounds on the rise.

Consider the problem then, of world leaders meeting to determine an acceptable sea

level rise, and using the probabilistic model and associated target temperature rise to meet it. Because of the uncertainty in the model, no matter what target temperature the world leaders choose, there will always be a chance that the target sea level will be exceeded.

One approach that the world leaders could take is to agree not only on a target sea level rise, but on an acceptable level of risk that the target will be exceeded. From this risk, they can then choose a target temperature rise to meet it. It is this sort of decision making that our quantile estimation algorithms enable.

Apart from the decision making application, quantile statistics are also useful as estimators in a regression context. The median for instance, is less sensitive to outliers than the mean [68], and can be semantically more meaningful than the mode when the underlying distribution is highly non-Gaussian or multi-modal.

4.2. Contributions

This chapter focuses on the problem of estimating cumulative and quantile statistics from a posterior probability embedding that is the result of an application of kernel Bayes' rule (KBR). These statistics are critical in performing risk-based decision making, and as estimators for non-Gaussian or multi-modal distributions. To address this problem, we present the following contributions:

Novel methods for computing Bayesian conditional cumulative estimates from KBR. We introduce a technique to recover estimates of the cumulative distribution function (CDF) of a reproducing kernel Hilbert space (RKHS) embedded probability distribution without first computing the density, by embedding an indicator function and utilising the reproducing property. This indicator embedding technique can recover a number of different CDF estimates with different computational costs and smoothness properties.

Extension of multi-modal kernel Bayes' rule to generate CDF estimates. We present a simple method of computing CDF estimates by integrating the kernel mixture posteriors from the multi-modal kernel Bayes' rule (MKBR) algorithm described in chapter three. These estimates have the advantage of being normalised and strictly non-decreasing, guaranteeing the estimate is a valid conditional distribution. We also examine the close relationship between the indicator embedding technique and this integration method.

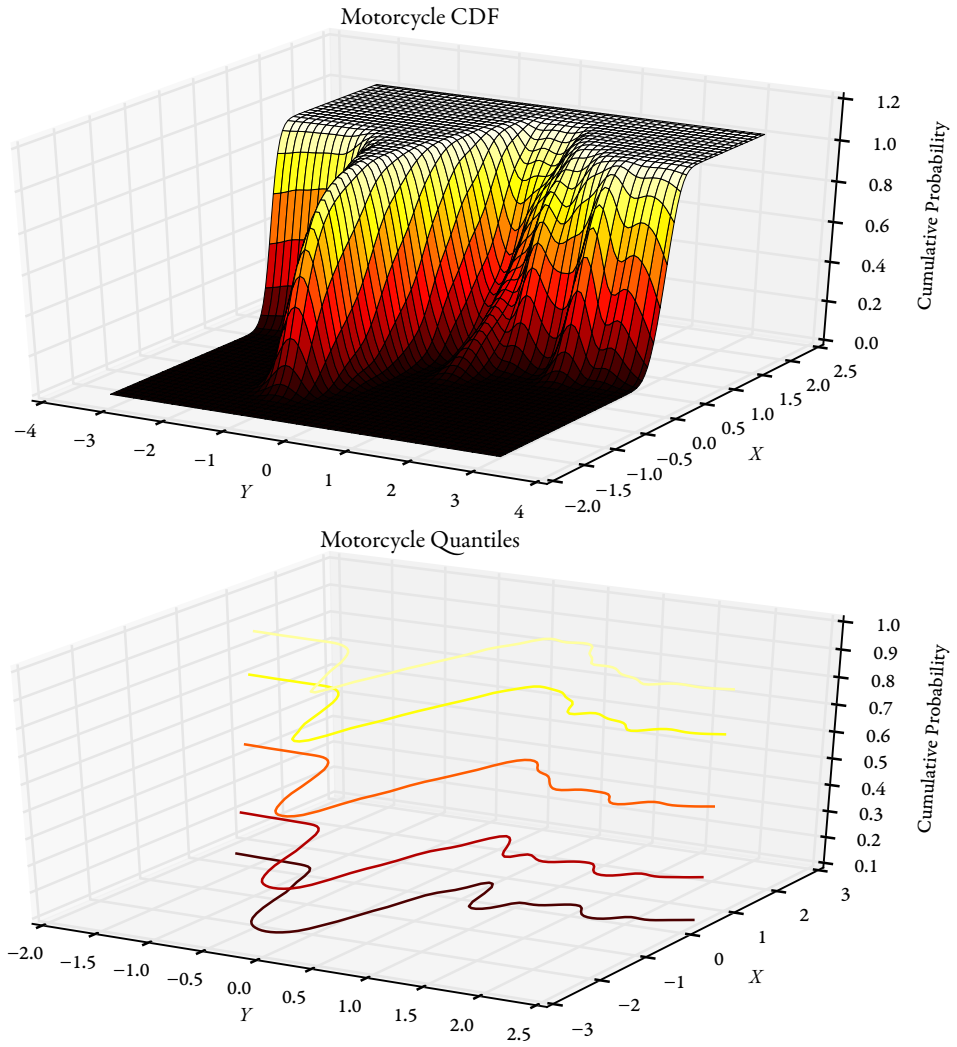


Figure 4.1.: (Top) An estimate of the conditional cumulative distribution function for the Motorcycle dataset created using the algorithms presented in this chapter. (Bottom) The 0.1, 0.25, 0.5 (median), 0.75 and 0.9 quantiles of the Motorcycle dataset. These curves can be thought of as level sets or contour lines of the cumulative distribution function above.

An algorithm for Bayesian conditional quantile estimation, based on the KBR CDF estimators. The conditional quantile itself can be estimated efficiently from a CDF using efficient one-dimensional root-finding techniques. We examine two optimisation strategies for MKBR parameter learning in this context: directly optimising error for a particular quantile, and using the negative log-probability (NLP) method in chapter three. The former demonstrates a potential increase in accuracy for a single given quantile, but must be re-trained when a different quantile is required. The NLP, on the other hand, is trained only once for any quantile estimate, and has the added advantage of ensuring non-crossing constraints on the quantile estimates.

Demonstration of competitive performance of our new quantile algorithm in a set of experiments. We compare the performance of the above techniques with other conditional quantile estimators in the literature. The experiments use standard machine learning datasets with complex multi-modal and non-linear behaviours. Our methods demonstrate competitive, and in some cases, superior performance to these algorithms. Our algorithms also make fewer assumptions about the underlying distributions.

4.3. Related Work

Quantile regression was first introduced by Koenker et. al. in 1978 [68]. This work considered the regression problem of inferring values of an unknown linear model given noisy samples of inputs and outputs. Rather than using the mean as an estimator of the function, Koenker chose the 0.5 quantile (the median), on the basis that it was less sensitive to outliers, and hence effective in cases with non-Gaussian or heavy tailed noise. A general regression quantile was then defined as the function value minimising the pinball loss [68](see section 4.4.3). For linear models, the minimisation was formulated as a linear programming problem and so solved efficiently.

The pursuit of conditional quantiles quickly divided the field into so-called discriminative methods, that attempt to directly compute the quantile from the training data through minimisation of the pinball loss, and generative methods, that attempt to first model the conditional cumulative distribution, and from there derive quantile estimates [67].

The discriminative initially demonstrated the advantage of flexibility— non-parametric

function estimation techniques could be applied that made no assumptions about the underlying distribution. Examples include locally-constant and locally-linear approximations [17, 127]. However, this flexibility also led to problems such as quantile crossing; in which a data point might be considered to be below the 0.5 quantile but above the 0.6 quantile [67].

An important development was the addition of non-crossing constraints to discriminative quantile regression solutions [46]. This ensured that different quantile estimates obeyed a strict ordering, with no two quantiles having the same function value. Such a strict ordering was achieved by adding a penalty term to the optimisation [20], or by reducing the class of possible functions used to represent the quantiles, for instance to classes of location-scale models [66, 46]. Unfortunately, such models implicitly restricted the underlying distribution, causing reduced applicability to data that did not fit the assumptions of the model [67].

Standard machine learning techniques have also been co-opted for conditional quantile estimation, including the support vector machine (SVM), which finds the decision boundary associated with the minimisation of the pinball loss, with additional non-crossing constraints [118]. However, in this algorithm, the non-crossing constraints may cause the resulting estimates to violate the (empirical) quantile definition.

More recently, Chernozhukov et. al. developed a monotisation procedure based on function re-arrangement to remove crossing from quantile estimates generated by other algorithms. The resulting quantiles were guaranteed to be more accurate, and no assumptions were made about the underlying distribution [19].

The 'quanting' algorithm [73] introduced an important development in discriminative estimation of conditional quantiles, which was to re-cast quantile regression as a classification problem. By placing a set of classifiers over the range of the regression, each classifier can be trained on whether the quantile is above or below it. The quantile estimate is then the expectation of this assignment, over all the classifiers. A set of importance weights were learned to minimise the error of the estimation.

Generative models for conditional quantile regression have also been examined in the semi-parametric and non-parametric setting. Linear models for Bayesian quantile estimation were examined in [128], using Laplacian likelihood functions and uniform priors. These methods require Markov chain Monte Carlo (MCMC) integration to perform in-

ference, as well as making the uni-modal assumption. Similar semi-parametric methods such as [48, 49] use Dirichlet process priors, which again required costly inference approximations.

Kernel conditional quantiles using Gaussian process regression to explicitly compute the cumulative distribution were examined in [94]. The underlying PDF was estimated using a Gaussian process, which enforced the non-crossing constraint and allowed for efficient inference. Additionally, heteroskedastic covariance functions were employed to account for input-dependent noise in the data. One limitation of the work was that the underlying Gaussianity assumption restricts applicability to uni-modal distributions.

A more general formulation of Bayesian quantile regression was developed in [117], which used Dirichlet process mixture models to represent the underlying joint distribution. This allowed a very general class of distributions to be represented, but inference required expensive MCMC integration.

Our work aims to overcome limitations of both these methods– giving the flexibility of a mixture distribution representation for modelling multi-modal behaviour, with the efficient inference inherent in the MKBR algorithm. It is to this new algorithm we now turn our attention.

4.4. Background

Before moving into the contributions of this chapter, we will give a brief overview of probability measures and cumulative distributions, quantiles, and conditional quantile regression. We will also describe the pinball loss function [68], which is the cost function used to evaluate the quality of a conditional quantile regression when given only test samples from the associated joint distribution.

4.4.1. Probability Measures

In chapter three, our work focussed on estimating the probability density function $P(X|Y = y)$ of a random variable X defined on a set $\mathcal{X}$ (usually $\mathbb{R}^N$) and given an observation y of $Y \in \mathcal{Y}$. Rather than considering the likelihood at a point x, cumulative and quantile estimation requires computing the probability of x being inside a given set $\Omega \subseteq \mathcal{X}$. In a slight

abuse of notation, we denote the function that maps sets Ω to the probability enclosed by that set as the probability measure for $\mathcal{X}$.

Definition 4.4.1. For Ω in the power set of $\mathcal{X}$, the probability measure M is the function for which $M(\Omega) = P(x \in \Omega)$.

By definition, $M(\mathcal{X}) = 1$ and $M(\varnothing) = 0$ where $\varnothing$ is the empty set. If $\mathcal{X}$ has a total ordering, then X has a cumulative distribution function, which is defined as the function giving the probability X is less than a particular value.

Definition 4.4.2. Given a random variable $X \in \mathbb{R}^N$ with probability density function $P(X)$, the cumulative distribution function $C(\mathbf{a})$ for $\mathbf{a} \in \mathbb{R}^N$ is the function for which $C(\mathbf{a}) = P(X \leq \mathbf{a})$.

The CDF is a special case of $M(\Omega)$:

$$C(a) = M(\{x\} : x \leq a). \tag{4.1}$$

If we restrict $\mathcal{X}$ to $\mathbb{R}^N$ such that X has a probability density function (PDF) $P(X)$, then M and hence C can both be written in terms of $P(X)$:

$$M(\Omega) = \int_{\Omega} P(\mathbf{x})\, \mathrm{d}\mathbf{x}, \tag{4.2}$$

and for C,

$$C(\mathbf{a}) = \int_{\mathbf{x} \leq \mathbf{a}} P(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{4.3}$$

For the rest of this chapter, we will make the assumption that $\mathcal{X} = \mathbb{R}^N$ and hence that $P(X)$ and $C(X)$ exist.

4.4.2. Quantiles

The quantile function of a random variable measures the value of that variable bounding a certain fraction of the probability mass of the PDF. The τ quantile $q(\tau)$ of $P(X)$ gives the value of X such that the probability of X being less than $q(\tau)$ is τ.

Definition 4.4.3. Let $P(X)$ be a probability distribution over a random variable $X \in \mathcal{X}$. Let $C(x) = P(X \leq x)$. Then for $\tau \in (0, 1)$, the τ-quantile of P is the function

$$q(\tau) = \inf(x: C(X) \geq \tau). \tag{4.4}$$

For instance, the 0.9 quantile is the value of X we expect that there is a 90% chance X is below. A particularly useful quantile is the 0.5 quantile, also known as the median. The median is the "middle value", in the sense there is a 50% chance X will be above the median, and a 50% chance it will below the median. For a Gaussian distribution, the median is equal to the mean, but this is certainly not the case in general.

The extension to conditional distributions is straightforward. A conditional quantile is simply the quantile of the conditional distribution.

Definition 4.4.4. Let $P(X|Y=y)$ be a conditional probability distribution over a random variable $X \in \mathcal{X}$, $Y \in \mathcal{Y}$. The conditional quantile $q(\tau|Y=y)$ is defined as the quantile q of the conditional $P(X|Y=y)$.

Conditional quantile curves can also be thought of as contour lines or level sets of the conditional CDF. If the $C(X|Y)$ is viewed as a height map in X and Y, then the 0.5 quantile say is the $C(X|Y) = 0.5$ level set. This idea is illustrated in figure 4.1.

4.4.3. Quantile Regression

Quantile regression is the problem of estimating the quantile $q(\tau|Y=y)$ of a distribution $P(X|Y=y)$ given a set of samples $\{x, y\}_{i=1}^{N}$ from the associated joint distribution $P(X, Y)$. An example, using the techniques developed in this chapter, is illustrated in figure 4.2.

In order to determine the *optimal* estimate of q in the regression context, we need to determine a cost function which measures how close our estimate is to the true quantile. The function with this property is called the pinball loss [68].

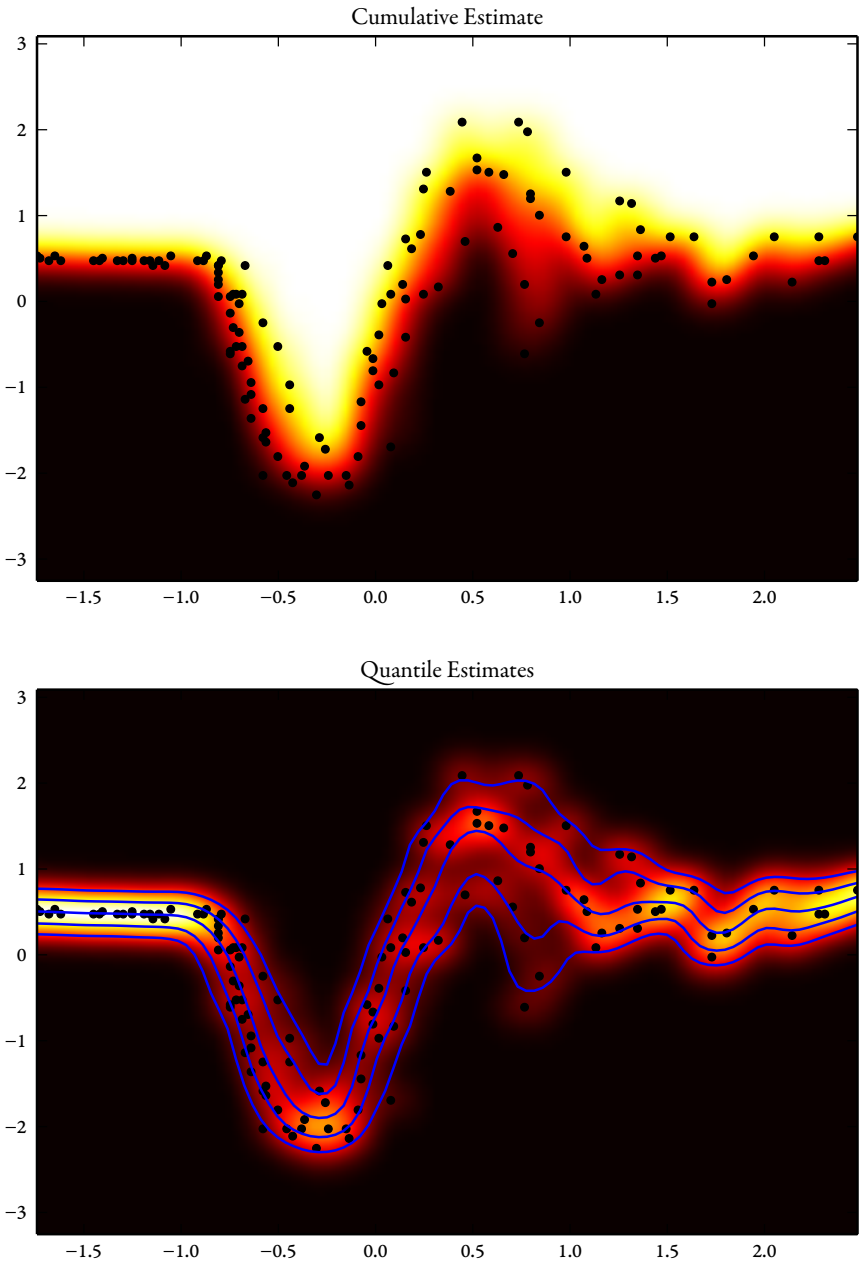


Figure 4.2.: Estimate of the cumulative embedding (top), and the quantiles 0.1, 0.25, 0.5, 0.75, 0.9 overlaying a PDF estimate (bottom)

Pinball Loss

The pinball loss is motivated by noting that the median, or 0.5 quantile of a distribution $P(X)$, minimises the expected absolute loss[68]:

$$q(0.5) = \underset{f}{\operatorname{argmin}} \, \mathrm{E}\left[|f - x|\right]. \tag{4.5}$$

Intuitively, the absolute loss is minimised because the median balances losses equally on either side. Consider two infinitesimal chunks of probability mass $\mathrm{d}x$ at locations x_1 and x_2 in the distribution $P(X)$. If the median is between these two points (if $x_1 \geq q(0.5) \geq x_2$), then the total absolute loss is

$$|q(0.5) - x_1| + |q(0.5) - x_2| = |x_1 - x_2| \tag{4.6}$$

and is independent of $q(0.5)$. If, however, the median is outside one of these points, then the absolute loss must be greater, by an amount equal to the distance q is from the nearest point:

$$|q(0.5) - x_1| + |q(0.5) - x_2| = |x_1 - x_2| + \min\left[|x_1 - q(0.5)|, |x_2 - q(0.5)|\right]. \tag{4.7}$$

Now, by definition, the median bisects the probability mass of the distribution. Therefore, for every infinitesimal volume of probability $x_1 \, \mathrm{d}x$ below the median, there will be a corresponding volume $x_2 \, \mathrm{d}x$ above the median and the integrand $|f - x|$ will always be minimised by the median, and hence the expected absolute loss. This argument can be generalised to consider the function that is minimised by an arbitrary quantile $q(\tau)$. We assert that it is the *pinball loss*:

Definition 4.4.5. Let $z = q(\tau)$ be the τ-quantile. The pinball loss is the function

$$L_\tau(x, z) = \begin{cases} (x - z)\tau & : x \geq z \\ (z - x)(1 - \tau) & : z > x \end{cases}. \tag{4.8}$$

To see that L_τ is indeed minimised by $q(\tau)$, consider again two points x_1 and x_2, with $x_1 \leq x_2$. Now though, we consider two infinitesimal probability masses $\mathrm{d}x_1 = \tau \, \mathrm{d}x$ and

$\mathrm{d}x_2 = (1-\tau)\,\mathrm{d}x$, with relative sizes $\mathrm{d}x_1/\,\mathrm{d}x_2 = \tau/(1-\tau)$. Now, if $q(\tau)$ is between x_1 and x_2, then the pairwise loss is

$$L_\tau(x_1, q))\,\mathrm{d}x_1 + L_\tau(x_2, q)\,\mathrm{d}x_2 = (q - x_1)(1-\tau)\tau\,\mathrm{d}x + (x_2 - q)(\tau)(1-\tau)\,\mathrm{d}x \tag{4.9}$$

$$= |x_1 - x_2|\,(\tau(1-\tau))\;\mathrm{d}x, \tag{4.10}$$

which is again, independent of $q(\tau)$. Any q outside the bounds between x_1 and x_2 will result in an additional term, just as in equation 4.7.

Now, critically, the ratio of probability mass below and above $q(\tau)$ is $\tau/(1-\tau)$, exactly the ratio of volumes of $\mathrm{d}x_1$ and $\mathrm{d}x_2$. Hence, every x_1 below $q(\tau)$ has a corresponding x_2 above, and so we can integrate over these pairs, resulting in the minimum of the expected pinball loss. This gives the motivation for using pinball loss to determine quantile estimates, but for a more rigorous proof of this result, see [68].

Unfortunately, in the regression context, we do not have access to the underlying distribution $P(X)$, but rather a set of training points $\{x_i, y_i\}_{i=1}^N$ from $P(X, Y)$. Therefore, given an estimate of the conditional quantile $q(\tau|Y=y)$, we can compute the empirical average of the expected pinball loss:

$$\bar{L}_\tau = \sum_{i=1}^{N} L_\tau\left(x_i, q(\tau|Y=y_i)\right). \tag{4.11}$$

The empirical pinball loss allows for training the quantile estimation algorithm, and also for evaluating the relative performance of different algorithms given a withheld testing set.

4.5. Measure Estimates from RKHS Embeddings

This section details novel methods for recovering an estimate of the measure function (and hence the CDF) from and embedded KBR posterior. Estimates of this function could be derived directly by integration (equations 4.2 and 4.3) using the MKBR posterior in chapter three. However, it is also possible to derive approximations for these functions directly from the embedded posterior $\mu[P(X)] \in H$. Direct approximations would increase the computational efficiency of the algorithm if a posterior estimate was not required.

To derive these direct approximations to C and M for a given embedding $\mu[P(X)] \in \mathcal{H}$ consider a function $f \in \mathcal{H}$. By the reproducing property:

$$\langle f, \mu[P(x)] \rangle = \mathrm{E}_P \left[f(x) \right] \tag{4.12}$$

$$= \int_{-\infty}^{\infty} f(x) P(x) \, \mathrm{d}x. \tag{4.13}$$

We can use this property to estimate M through embedding *indicator functions*. The indicator function $\chi_\Omega(x)$ is 1 when x is in a subspace of $\mathbb{R}^N$ denoted as Ω and zero otherwise.

Definition 4.5.1. The indicator function $\chi_\Omega \colon \mathbb{P}(\mathbb{R}^N) \to (0, 1)$ is the function in which for all $x \in \mathbb{R}^N$,

$$\chi_\Omega(x) = \begin{cases} 1 & : x \in \Omega \\ 0 & : x \notin \Omega \end{cases}. \tag{4.14}$$

where $\mathbb{P}(\mathbb{R}^N)$ is the power set of $\mathbb{R}^N$.

Let χ_Ω be an element of $\mathcal{H}$, then by the reproducing property, it is possible to recover the probability measure function from an embedded distribution $\mu[P(X)]$ by taking the dot product,

$$\langle \chi_\Omega, \mu[P(x)] \rangle = \mathrm{E} \left[\chi_\Omega \right] \tag{4.15}$$

$$= \int_{\mathcal{X}} \chi_\Omega(x) P(x) \, \mathrm{d}x \tag{4.16}$$

$$= \int_{\Omega} P(x) \, \mathrm{d}x \tag{4.17}$$

$$= M(\Omega). \tag{4.18}$$

Unfortunately, empirical estimates of KBR created from a finite set of samples $\{x_i\}_{i=1}^N$ exist in an RKHS that will not in general contain χ_Ω. Only functions in the familiar form $\sum_{i=1}^N \alpha_i k(x_i, \cdot)$ are admitted. However, a straightforward optimisation allows us to deter-

mine the optimal embedding of a function f in terms of a limited basis [105], and so compute the empirical approximation of χ_Ω.

4.5.1. Optimal Approximate Embeddings

For the moment, assume f is an arbitrary function in a RKHS H. Given a basis induced by a set of points $\{x_i\}_{i=1}^N$, then finding an approximate f in the span of this basis is equivalent to determining optimal weights $\boldsymbol{\alpha}$ such that

$$\hat{f} = \sum_{i=1}^{N} \alpha_i k(x_i, \cdot). \tag{4.19}$$

The optimal weights $\boldsymbol{\alpha}^*$ minimise the distance between f and $\hat{f}$ in the Hilbert norm sense. Therefore,

$$\begin{aligned}
\boldsymbol{\alpha}^* &= \operatorname*{argmin}_{\alpha \in \mathbb{R}^N} \|f - \hat{f}\| && (4.20)\\
&= \operatorname*{argmin}_{\alpha \in \mathbb{R}^N} \langle f, f\rangle - 2\langle f, \hat{f}\rangle + \langle \hat{f}, \hat{f}\rangle && (4.21)\\
&= \operatorname*{argmin}_{\alpha \in \mathbb{R}^N} \langle f, f\rangle - 2\boldsymbol{\alpha}^T \mathbf{f}[X] + \boldsymbol{\alpha}^T G_{XX} \boldsymbol{\alpha}, && (4.22)
\end{aligned}$$

where $(\mathbf{f}[X])_i = f(x_i)$. Differentiating this expression and setting the result to zero yields the optimal weights

$$\boldsymbol{\alpha} = G_{XX}^{-1} \mathbf{f}[X]. \tag{4.23}$$

4.5.2. Optimal Empirical Measure Estimate

Using equation 4.23, we can now embed the indicator function χ_Ω into a RKHS H defined by a kernel k and a set of points $\{X_i\}_{i=1}^N$. The approximation is

$$\hat{\chi}_\Omega = G_{XX}^{-1} \boldsymbol{\chi}_\Omega[X], \tag{4.24}$$

where $(\boldsymbol{\chi}_\Omega)_i = \chi_\Omega(X_i)$. Hence, for an embedding $\mu[P(X)] = \mathbf{k}_X[\cdot]^T\boldsymbol{\beta}$, the estimate of the probability volume function $M(\Omega)$ is

$$M(\Omega) = \langle \hat{\chi}_\Omega, \mu[P(X)] \rangle \tag{4.25}$$
$$= (G_{XX}^{-1}\mathbf{f}[X])^T G_{XX}\boldsymbol{\beta} \tag{4.26}$$
$$= \boldsymbol{\chi}_\Omega[X]^T\boldsymbol{\beta}, \tag{4.27}$$

where $(\boldsymbol{\chi}_\Omega[X])_i = \chi_\Omega(x_i)$. This now gives us a fast and efficient way to compute the probability measure function $M(\Omega)$ by a simple dot product of coefficient vectors in $\mathbb{R}^N$.

Cumulative Estimation

For a cumulative estimate, equation 4.25 simplifies to a sum of weights:

$$C(a) \approx \sum_{\{i\}:x_i<a} \beta_i. \tag{4.28}$$

A similar expression for the empirical estimate of the cumulative distribution from $\mu[P(X)]$ was developed independently by Kenji Fukumizu [30].

Example

Figure 4.3 illustrates an approximation to the CDF of a posterior embedding calculated using KBR and the Motorcycle dataset. Note the weight vector is graphed in the left of the figure, from which the cumulative sum is generated on the right.

One thing immediately noticeable about this approximation to the CDF is that it is piecewise constant, with a step change occurring at every training point. Additionally, because the KBR output contains weights which can be negative, it is also not strictly non-decreasing. We address the smoothness issue first, in the next section.

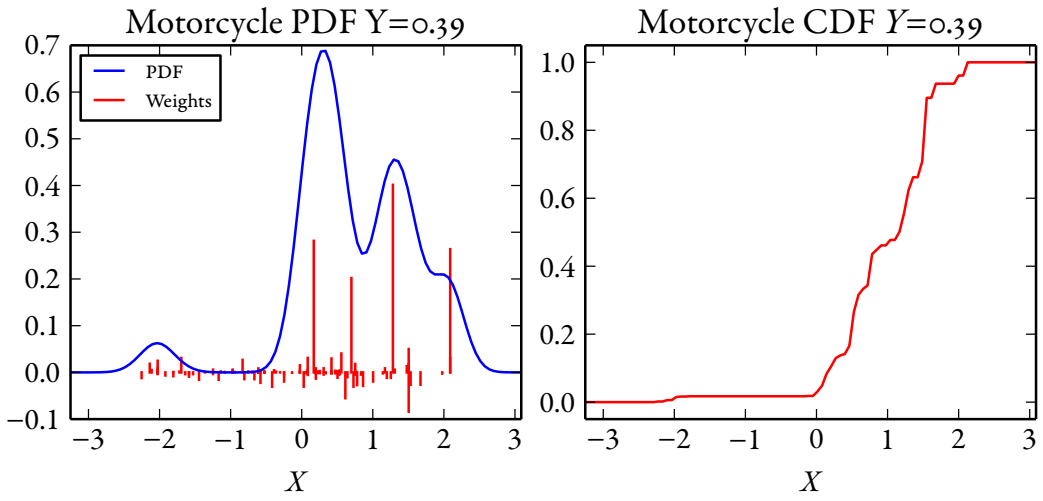


Figure 4.3.: (Left) A MKBR estimate of the PDF from the Motorcycle dataset. The blue line indicates the PDF, and the red bars indicate the weights of training points at those X locations. (Right) The resulting CDF estimate using equation 4.28. Note that the estimate jumps at locations corresponding to training points.

4.6. Smooth Measure Approximations

A smooth estimate of the measure function, and hence the CDF, would have some desirable properties over the approximation in the previous section. First, if the underlying distribution was smooth then the estimate could approximate it more accurately. The estimate would also be differentiable, allowing for more efficient root-finding methods to be used for quantile estimation.

One way to achieve a smooth estimate of the measure function M is to approximate the indicator function as

$$\hat{\chi}_\Omega = \mu[\chi_\Omega] \tag{4.29}$$

$$= \int_\Omega k(x, \cdot)\, \mathrm{d}x. \tag{4.30}$$

Clearly, $\hat{\chi}_\Omega$ converges to χ_Ω in the limit as the kernel bandwidth goes to zero. This convergence property makes intuitive sense, as it follows the convergence of the KBR itself,

which expects a vanishing kernel bandwidth as the number of samples in the empirical estimate goes to infinity [34].

To use the same reproducing property approach as in equation 4.25, $\hat{\chi}$ needs to be re-written in terms of the empirical RKHS $\mathcal{H}$ induced by KBR on a set of training points $\{x_i\}_{i=1}^N$. Using the optimal approximate embeddings in equation 4.23,

$$\mu[\Omega] = G_{XX}^{-1}\hat{\boldsymbol{\omega}}[X], \tag{4.31}$$

where $(\hat{\boldsymbol{\omega}}[X])_i = \int_\Omega k(x, x_i)\,\mathrm{d}x$. Therefore, the estimate for the probability measure $M(X)$ of an arbitrary distribution $P(X)$ with embedding $\mu[P(X)] = \mathbf{k}_X[\cdot]^T\boldsymbol{\alpha}$ is

$$M(X) = \langle \mu[P(X)], \hat{\boldsymbol{\omega}}[X] \rangle \tag{4.32}$$

$$= \boldsymbol{\alpha}^T\hat{\boldsymbol{\omega}}[X]. \tag{4.33}$$

4.6.1. Example: Cumulative distribution

To estimate the cumulative distribution $C(a)$ the vector $\hat{\boldsymbol{\omega}}[X]$ is

$$\hat{\boldsymbol{\omega}}[X]\colon\ \omega_i = \int_{x \leq a} k(x, x_i)\,\mathrm{d}x. \tag{4.34}$$

An illustration of the same Motorcycle dataset posterior with this smooth cumulative estimate is given in figure 4.4. In this example, the kernel was a one-dimensional Gaussian, which means the expression for the cumulative estimate is

$$C(a) = \sum_{i=1}^N \alpha_i\omega_i \tag{4.35}$$

$$= \sum_{i=1}^N \alpha_i \int_{x \leq a} \exp\left[\frac{||x - x_i||^2}{2\sigma^2}\right]\mathrm{d}x \tag{4.36}$$

$$= \frac{1}{2}\sum_{i=1}^N \alpha_i\left[1 + \operatorname{erf}\left(\frac{a - x_i}{\sigma\sqrt{2}}\right)\right]. \tag{4.37}$$

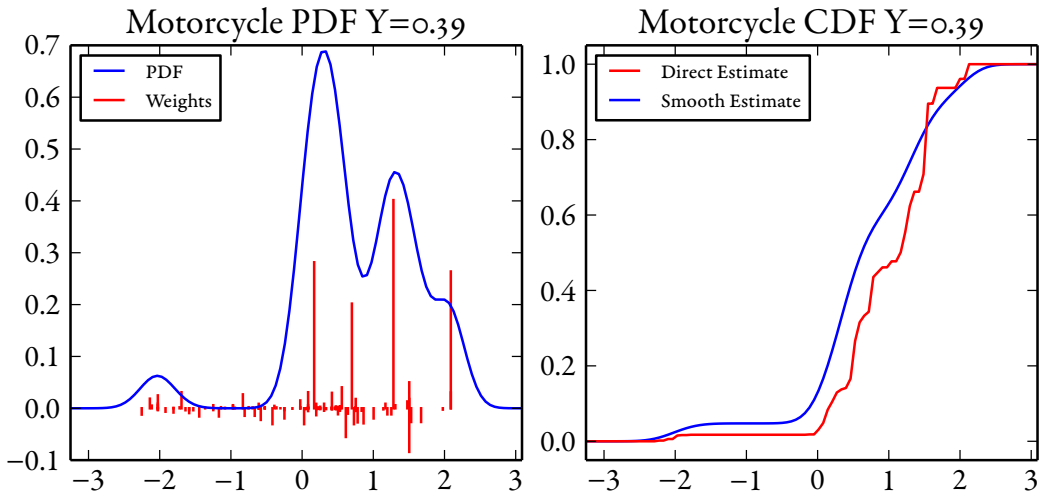


Figure 4.4.: (Left) A MKBR estimate of the PDF from the Motorcycle dataset. The blue line indicates the PDF, and the red bars indicate the weights of training points at those X locations. (Right) The resulting smooth CDF estimate using equation 4.34 overlaid on the previous (non-smooth) estimate.

Though the cumulative estimate is now smooth, because of the negative weights (visible in figure 4.5), it is still not guaranteed to be strictly non-decreasing. To rectify this, we examine the relationship between the cumulative estimations we have derived, and the probability density estimations in chapter three.

4.7. CDF-From-PDF Estimation Techniques

Given the MKBR's ability to recover PDF estimates from RKHS embeddings, directly integrating the PDF is also an attractive choice for measure and CDF estimation. Such an estimate would be both smooth and strictly non-decreasing, and hence be a valid CDF unconditionally.

Given a set of training points $\{x_i, y_i\}_{i=1}^N$ with kernel k, the posterior estimates generated

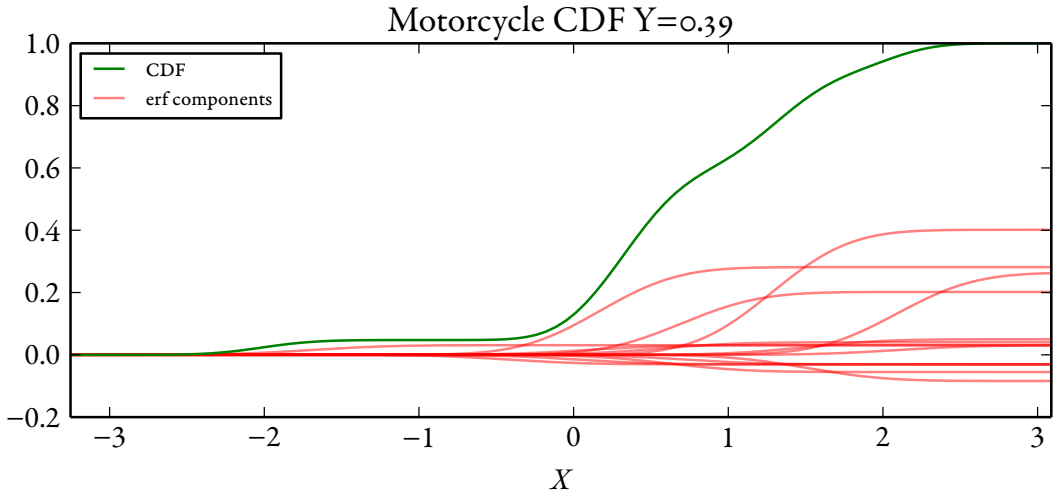


Figure 4.5.: CDF estimate of the Motorcycle data with a Gaussian kernel. The individual components from equation 4.35 are also plotted. Just as the posterior density is a sum of Gaussians, the posterior cumulative is a sum of error functions. Note, negative weights mean that some of the contributions from the terms in the sum are negative. The resultant CDF estimate may not be strictly non-decreasing.

by MKBR are parametrised as a mixture distribution of kernels,

$$P(X|Y=y) = \sum_{i=1}^{M} w_i k(\mu_i, \cdot). \tag{4.38}$$

Therefore, the measure function for some set $\Omega \subseteq \mathcal{X}$ is the integral of the PDF,

$$M(\Omega|Y=y) = \sum_{i=1}^{M} w_i \int_{\Omega} k(\mu_i, x)\,\mathrm{d}x. \tag{4.39}$$

For a cumulative distribution estimate, this equates to

$$C(a|Y=y) = \sum_{i=1}^{M} w_i \int_{x \leq a} k(\mu_i, x)\,\mathrm{d}x. \tag{4.40}$$

For many kernels, these integrals have an analytic expression. Even in the event an analytic solution does not exist, the integral is independent of the posterior weights, and hence approximations to the kernel integrals can be pre-computed before inference begins.

One kernel for which the integral is well-known is the Gaussian kernel. Assuming the bandwidth of the mixture components is equal to the kernel bandwidth of KBR, and the mixtures are centred on the training points, the cumulative estimate reduces to

$$C(a|Y{=}y) = \frac{1}{2}\sum_{i=1}^{M} w_i \left[1 + \operatorname{erf}\left(\frac{a - x_i}{\sigma\sqrt{2}}\right)\right]. \tag{4.41}$$

This is identical to the smooth cumulative derived in equation 4.35, but with the posterior weights $\mathbf{w}$ (equation 3.19) replacing the embedding weights (equation 2.96). Because the posterior weights are guaranteed to be positive, the resulting CDF estimate is guaranteed to be non-decreasing. This equivalence also highlights another interpretation of the smooth CDF estimate generated in section 4.6.1. It is simply the integral of the normalised posterior embedding.

4.8. Overview of CDF Estimation Algorithms

The first two techniques we have presented for CDF estimation are based on the direct embedding of an indicator function to compute the integral of the posterior density via the reproducing property. The use of a simple piecewise-constant indicator function results in equation 4.28. This CDF estimate is piecewise constant, changing value only at the location of training points. It is also not guaranteed to be a valid CDF: the estimate may not be strictly non-decreasing.If we instead convolve the indicator function with the kernel, we retrieve the smooth approximation to the cumulative distribution seen in equation 4.34. This estimate is also not guaranteed to be strictly non-decreasing.

The other two techniques we have presented for CDF estimation are based on the PDF estimation techniques presented in chapter three. In that chapter we demonstrated how to recover a posterior PDF from the RKHS embedding output by kernel Bayes' rule using one of two pre-image techniques; a full optimisation of the posterior weights (equation 3.19), and an estimate based on a positive normalised embedding (equation 3.39).

Therefore, we also have two corresponding cumulative estimates based on integration of these PDFs. Both methods return normalised, strictly non-decreasing conditional cumulative estimates.

4.9. KBR Quantile Estimation

The generative approach for estimating a quantile is first to estimate the cumulative density function $C(x)$ and from this solve the equation $C(x) = \tau$ for a quantile τ. Combining the cumulative estimates derived in section 4.8 with a numerical method to find the root of the expression $C(X) - \tau$ will yield quantile estimates. However, the question then becomes how to train the free parameters in the KBR and MKBR algorithms required to generate the CDF estimates.

4.9.1. Parameter Learning

We have considered two approaches to optimising the free parameters in our KBR quantile estimators, which are described in detail in section 3.6. The first is simply to use the pinball loss described in section 4.4.3, evaluated over a leave-out set of the training data generated from a cross-validation scheme. For a particular quantile, a function which minimises the pinball loss will also minimise the error in the quantile estimate. Additionally, the loss function is tolerant to CDF estimates which may not be strictly non-decreasing. The disadvantage of this cost function is that it is specific to a particular quantile. Therefore, estimating different quantiles from the same training data requires different optimisations. The result is that quantile estimates may actually cross for certain query locations.

Another possible approach to parameter learning is to optimise the parameters with respect to the PDF that is being used to generate the CDF estimate. We can utilise the same cost function as in chapter three (section 3.6), the NLP of a leave-out set of the training data. The NLP approach has a significant advantage, in that it is independent of the particular quantile estimate chosen. This means that a single CDF is generated for all quantile queries, and therefore different quantiles are guaranteed not to cross. Obviously this approach could not be used when we have directly estimated the quantile as in sections 4.5.2 and 4.6.1.

4.9.2. Summary of Algorithms

Given a kernel k, a set of training samples $\{x_i, y_i\}_{i=1}^N$ from a joint distribution $P(X, Y)$, weighted samples $\{u_i, \gamma_i\}_{i=1}^M$ from the prior $P(X)$, and an observation y, KBR yields a posterior embedding $\mu[P(X|Y=y)] = \mathbf{k}_X[\cdot]^T\boldsymbol{\alpha}$ in the RKHS H with $\boldsymbol{\alpha} \in \mathbb{R}^N$. As we have described four methods for estimating the conditional cumulative distribution, there are four associated conditional quantile estimates. The first uses the direct CDF approximation 4.28.

Definition 4.9.1 (Direct Embedding Quantile Estimator)**.** The direct embedding τ-quantile estimate q is

$$q(\tau) = x\colon \frac{1}{\|\boldsymbol{\alpha}\|} \sum_{\{i\}:x_i<x} \alpha_i = \tau. \tag{4.42}$$

For the direct embedding quantile we minimise the pinball loss (equation 4.11) of the training data using k-fold cross-validation over the free parameter vector θ as described in section 3.6. We denote this algorithm as DR.

Using the smooth approximation of the CDF given in equation 4.34 results in another direct embedding method:

Definition 4.9.2 (Smooth Embedding Quantile Estimator)**.** The smooth embedding τ-quantile estimate q is

$$q(\tau) = x\colon \frac{1}{\|\boldsymbol{\alpha}\|}\boldsymbol{\alpha}^T\hat{\mathbf{K}}_q[X] = \tau, \tag{4.43}$$

where $(\hat{\mathbf{K}}_q[X])_i = \int_{-\infty}^{q} k(x_i, x)\,\mathrm{d}x$. We train this algorithm by minimising the pinball loss (equation 4.11) using k-fold cross validation over the free parameter vector θ as described in section 3.6. We denote this algorithm as DS.

Integration of a PDF estimate $P(X|Y=y)$ obtained through a pre-image optimisation described in section 3.5.3 yields the pre-image quantile estimator:

Definition 4.9.3 (Pre-Image Quantile Estimator)**.** The pre-image posterior τ-quantile estimate q is

$$q(\tau) = x\colon\ \mathbf{w}^T\hat{\mathbf{K}}_q[X] = \tau, \tag{4.44}$$

where $(\hat{\mathbf{K}}_q[X])_i = \int_{-\infty}^{q} k(x_i, x)\,\mathrm{d}x$ and $\mathbf{w}$ are the weights of the posterior estimate $P(X|Y = y) = \sum_{i=1}^{N} w_i k(x_i, \cdot)$ generated from equation 3.19. We can train the MKBR parameter vector θ (see section 3.6) for this algorithm using the pinball loss (equation 4.11) or the NLP (equation 3.41) using the k-fold cross-validation scheme described in section 3.6. We denote this algorithm as JB when trained with pinball loss, and JL when trained with negative log-probability.

Finally, integration of a PDF estimate obtained through a normed weights method described in section 3.5.4 yields the normed weights quantile estimator:

Definition 4.9.4 (Normed Weights Quantile Estimator)**.** The norm posterior τ-quantile estimate q is

$$q(\tau) = x\colon\ \mathbf{w}^T\hat{\mathbf{K}}_q[X] = \tau, \tag{4.45}$$

where $(\hat{\mathbf{K}}_q[X])_i = \int_{-\infty}^{q} k(x_i, x)\,\mathrm{d}x$ and $\mathbf{w}$ are the weights of the posterior estimate $P(X|Y = y) = \sum_{i=1}^{N} w_i k(x_i, \cdot)$ from equation 3.39. Like the pre-image quantile estimator, we can train the MKBR parameters for this algorithm using the pinball loss (equation 4.11) or the NLP (equation 3.41) using the k-fold cross-validation scheme described in section 3.6 in both cases. We denote this algorithm as NB when trained with pinball loss, and NL when trained with negative log-probability.

Table 4.1 lists the properties of each of these algorithms, including whether the underlying CDF estimate is smooth or non-decreasing, and whether the algorithm enforces the non-crossing property. It also describes the time complexity of each algorithm, as detailed in the next section.

Algorithm	Smooth	Non-Decreasing	Non-Crossing	Complexity
DR				$O(N\log N)$
DS	✓			$O(N\log N)$
NB	✓	✓		$O(N\log N)$
JB	✓	✓		$O(N^3\log N)$
NL	✓	✓	✓	$O(N\log N)$
JL	✓	✓	✓	$O(N^3\log N)$

Table 4.1.: Comparison of Quantile estimation techniques. For the computational complexity bound, N is the number of training points.

Computational Complexity

The computational complexity of these algorithms varies from $O(N\log N)$ to $O(N^3\log N)$ in the number of training points N. The 1-D root-finding required to solve the quantile equations adds a factor of $\log N$ in all cases (using a binary search), the difference comes from the cumulative estimation algorithms.

The cheapest cumulative evaluate is the direct embedding estimator (DR)— this is simply a (conditional) sum of the mixture weights output by KBR and is therefore $O(N)$ in the number of training points. The smooth embedding estimator (DS) and the normed weights estimator (NL and NB) are also inexpensive to compute, provided an efficient estimate of the kernel integral exists. They are also $O(N)$, but with a larger constant due to the kernel integration. The most expensive are the pre-image estimators (JL and JB), which require solving a quadratic program to determine the mixture weights and are therefore $O(N^3)$. This is only relevant if a PDF estimate was not required, and the pre-image was computed only for the quantiles. If a pre-image estimate is already available then the JL and JB are also $O(N)$.

4.10. Experiments

We now test the performance of our quantile estimators on a number of standard machine learning datasets from the literature. We compare our algorithms to state-of-the-art generative and discriminative techniques for conditional quantile estimation.

We evaluated the algorithms on four datasets commonly used in the literature for conditional quantile estimation tests: Antigen, Weather, Bone Mineral Density and Motorcycle. The Antigen dataset samples the concentration of various molecules [94] in a blood sample. The dataset contains 97 points, with eight-dimensional observations Y and a one-dimensional state X. The Weather dataset measures a meteorological variable distributed over the surface of the Earth. It has two-dimensional observations Y and a one-dimensional state X, and contains 238 points. The Motorcycle dataset is a record of measurements taken from an accelerometer during a front-on motorcycle collision, and is often used because it contains heteroskedastic, non-Gaussian behaviour. The dataset contains 137 points, with a single input dimension Y (time), and a single output dimension X. Finally, the Bone Mineral Density dataset contains relative spinal bone mineral density measurements from North American adolescents of various ages. It has a single input variable Y (age), a single output variable X (bone mineral density), and 485 points. The original data were recorded in [7]. Antigen, Weather and Motorcycle were taken from the UCI repository [6] and Bone Mass Density from the "ElemStatLearn" R package [45].

For all experiments, performance was evaluated by five-fold cross validation and the average pinball loss over the hold-out set. The average and standard deviation of the performance over these five tests was used as the final score. Three quantiles were tested for each dataset; 0.1, 0.5 and 0.9. Any categorical variables in the data were ignored. All X and Y variables were scaled to have 0 mean and unit variance in each dimension.

4.10.1. Our Algorithms

We test the six quantile estimation algorithms in section 4.9. These are the direct embedding quantile estimator (DR), the smooth embedding estimator (DS), the pre-image quantile estimator with pinball loss learning (JB) and negative log-probability learning (JL), and the normed weights quantile estimator with pinball (NB) and negative log-probability learning (NL).

For all KBR-based algorithms, we used a Gaussian kernel for both the query variables Y and the output variable X. For the query variables Y, we parametrised the kernel with a length scale for every dimension, and whitened them with a rotation matrix that diagonalised their covariance. This implemented a form of automatic relevance determina-

tion [98]. All our algorithms were trained with (nested) five-fold cross-validation.

4.10.2. Comparisons

The comparison results for these algorithms were taken from the results in [94]. The following paragraphs give a brief overview of each of the algorithms against which we compared. For more detail, see the stated reference, or section 4.3.

Linear Quantile Estimation This is the linear discriminative quantile estimator described in [68]. We denote it with the letter A in our tabulation of results.

Quantile SVM This is the non-parametric estimator of the quantile based on dual optimisation of the loss function through a support vector machine, as documented in [118]. The RBF kernel was used for this algorithm, with kernel width and regularisation fitted as described in [94]. We denote this algorithm with the letter B.

Reduction to Classification This is the algorithm for reducing a quantile estimation problem to a series of classifiers [73]. The results from this algorithm are taken from [94], which used 100 GP classifiers trained with expectation propagation and an RBF kernel. We denote this algorithm with the letter C.

Quantile GP This algorithm is the Gaussian-processed based direct estimate of the cumulative distribution [94]. An RBF kernel was used, with the hyper-parameters learned by optimising the marginal log-likelihood. We denote this algorithm with the letter D.

Heteroskedastic Quantile GP This algorithm is the same as the Quantile GP, except that the kernel is a heteroskedastic RBF with 10 latent pseudo-observations controlling the bandwidth along the length of the input dimension [94]. This algorithm is denoted with the letter E.

4.10.3. Results

The average pinball loss for each algorithm in the three experiments is given in figure 4.6. Overall, all algorithms that obtain a quantile estimate through PDF estimation performed

as well if not better than the algorithms from the literature. The two best performing algorithms were *NB* and *JB*, which are the normed-weights and pre-image methods, both using the pinball loss cost function for training. The fact that this cost function performed better than the NLP is unsurprising, given that the training was optimising the same function that would evaluate the algorithm's performance.

This demonstrates a trade-off between using pinball loss and receiving slightly better performance, or using negative log-probability (NL and JL) and ensuring the non-crossing constraint holds. The NL and JL algorithms were still competitive, but also beaten out by the kernel conditional quantile (D).

With both the NLP and pinball loss functions, there was little difference on average between the pre-image estimator and the normed-weights estimator. This fact in particular is worth noting when considering a real-time application of these algorithms, as the pre-image estimator has a much greater computational cost.

Over the three experiments, we see that extremal quantiles 0.1 and 0.9 are significantly lower pinball loss compared to the median (0.5). There appear to be no algorithm that is especially capable at a particular quantile, although two direct methods (DR and DS) are much more competitive with the extrema than the median. The likely explanation is that, in general, an accurate (empirical) estimate of the median is affected by nearby points both above and below it. These direct methods however only sum contributions from points lying below the median. The posterior methods on the other hand, sum contributions from mixture components centred at all training points. Far away from the median, there are fewer points nearby and so the effect is reduced.

The full results for each individual experiment and quantile are also tabulated. For Antigen, this information is given in table 4.2. Curiously, the three top performers for the quantiles were each from a totally different family of algorithms; our smooth direct embedding estimator (DS), the quantile SVM (B), and the quantile GP (D). The full results for the Weather, and Bone Mineral Density tests are given in tables 4.3 and 4.5.

From the tabulated results in tables 4.4 and 4.5, we can see that algorithm E matched or exceeded all other algorithms in Motorcycle dataset, and was competitive but did not beat our algorithms in the Bone Mineral Density dataset. The likely explanation is the strong heteroskedasicity of the Motorcycle dataset benefited from a modelling approach that explicitly took it into account. This raises a possible avenue of future work— including

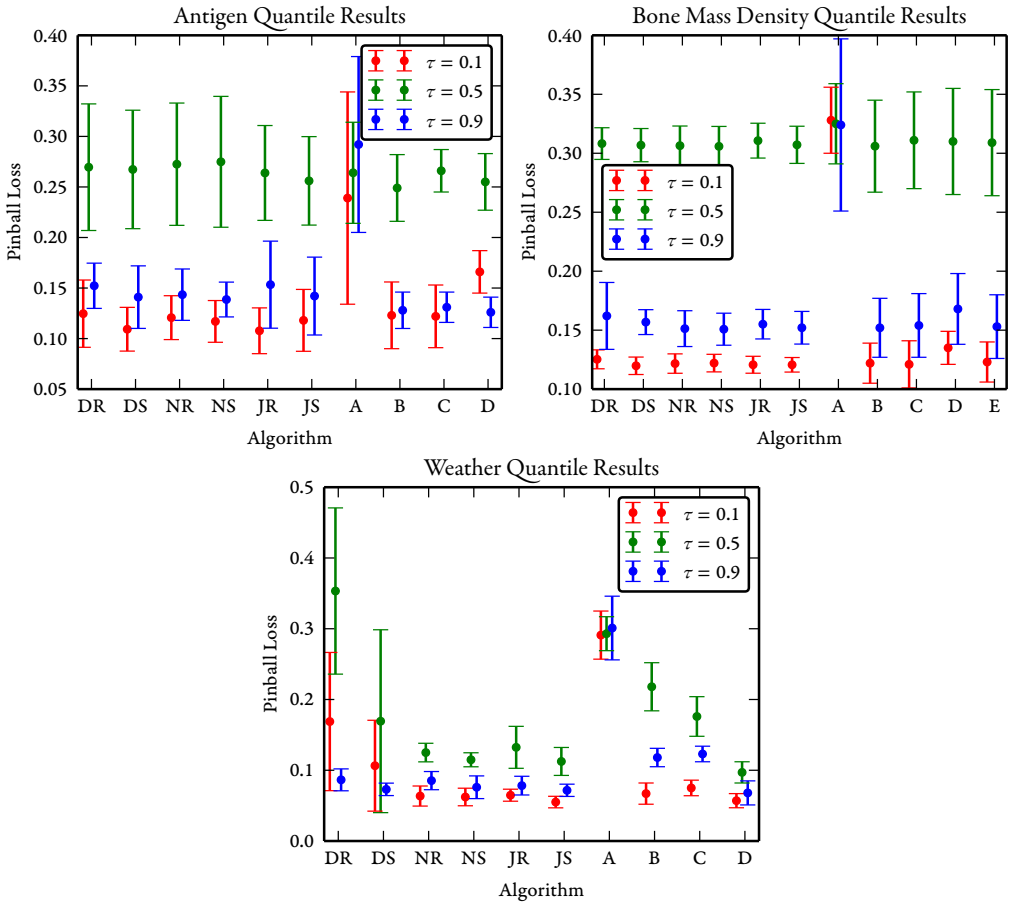


Figure 4.6.: Results from the three quantile experiments. The error bars in these figures represent ±1 standard deviation of the pinball loss over the testing set.

heteroskedastic kernel in our KBR based methods to improve performance.

Figure 4.7 plots the quantiles estimated for the Motorcycle dataset for three of the algorithms. The piecewise-constant quantile estimates are clearly visible in the DR plot. The JB plot has each quantile trained separately, and as a result the 0.1 and 0.9 quantile both fit the boundary of the data very closely. The NL plot shows a much smoother representation, in line with the fact that these are quantiles for a single underlying cumulative distribution (trained on the NLP).

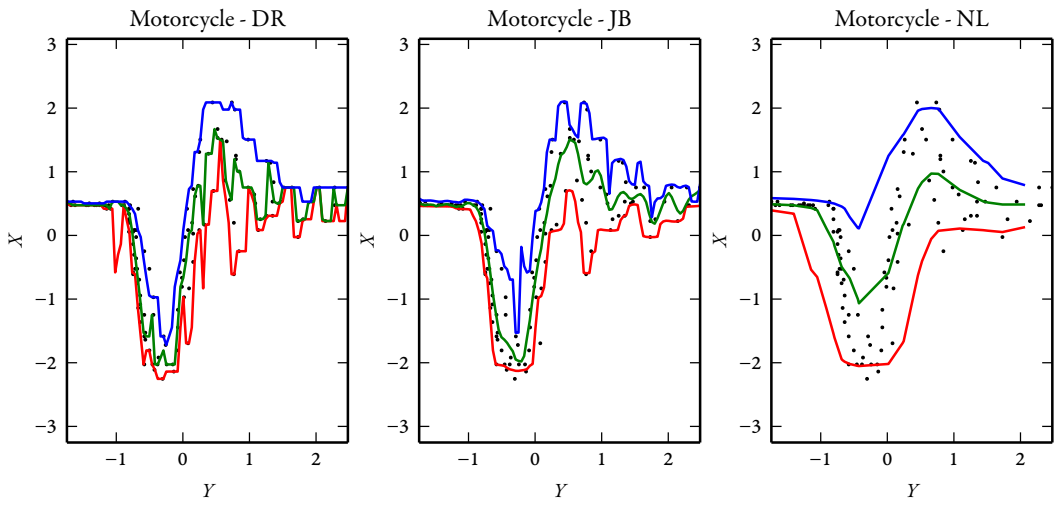


Figure 4.7.: Examples of 0.1 (red), 0.5 (green) and 0.9 (blue) quantile estimates for the Motorcycle dataset, for the DR (left), JB (centre) and NL (right) algorithms.

Method	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
DR	0.125 ± 0.033	0.270 ± 0.063	0.152 ± 0.022
DS	**0.109 ± 0.022**	0.267 ± 0.059	0.141 ± 0.031
NB	0.117 ± 0.021	0.275 ± 0.065	0.139 ± 0.017
JB	0.118 ± 0.031	0.256 ± 0.044	0.142 ± 0.039
NL	0.136 ± 0.035	0.294 ± 0.030	0.141 ± 0.014
JL	0.124 ± 0.017	0.260 ± 0.033	0.132 ± 0.022
A	0.239 ± 0.105	0.264 ± 0.050	0.292 ± 0.087
B	0.123 ± 0.033	**0.249 ± 0.033**	0.128 ± 0.018
C	0.122 ± 0.031	0.266 ± 0.021	0.131 ± 0.015
D	0.166 ± 0.021	0.255 ± 0.028	**0.126 ± 0.015**

Table 4.2.: Antigen results.

Method	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
DR	0.169 ± 0.098	0.353 ± 0.117	0.086 ± 0.015
DS	0.106 ± 0.064	0.169 ± 0.129	0.073 ± 0.009
NB	**0.062 ± 0.012**	0.115 ± 0.010	0.076 ± 0.016
JB	0.055 ± 0.008	0.112 ± 0.020	0.072 ± 0.009
NL	0.063 ± 0.008	0.143 ± 0.029	0.087 ± 0.016
JL	0.077 ± 0.012	0.136 ± 0.034	0.094 ± 0.016
A	0.291 ± 0.034	0.293 ± 0.024	0.301 ± 0.045
B	0.067 ± 0.015	0.218 ± 0.034	0.118 ± 0.013
C	0.075 ± 0.011	0.176 ± 0.028	0.123 ± 0.011
D	0.057 ± 0.010	**0.097 ± 0.015**	**0.068 ± 0.017**

Table 4.3.: Weather results.

Method	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
DR	0.163 ± 0.060	0.240 ± 0.037	0.096 ± 0.024
DS	0.180 ± 0.075	0.465 ± 0.181	0.120 ± 0.022
NB	0.096 ± 0.019	0.194 ± 0.033	0.092 ± 0.015
JB	0.101 ± 0.030	0.204 ± 0.022	0.107 ± 0.041
NL	0.097 ± 0.016	0.210 ± 0.034	0.092 ± 0.016
JL	0.091 ± 0.007	0.199 ± 0.032	0.091 ± 0.010
A	0.396 ± 0.080	0.389 ± 0.019	0.387 ± 0.056
B	0.090 ± 0.012	0.202 ± 0.019	0.085 ± 0.008
C	0.094 ± 0.011	0.190 ± 0.015	0.083 ± 0.010
D	0.092 ± 0.025	**0.186 ± 0.018**	0.089 ± 0.010
E	**0.079 ± 0.019**	0.187 ± 0.021	**0.070 ± 0.016**

Table 4.4.: Motorcycle results.

Method	$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.9$
DR	0.125 ± 0.008	0.308 ± 0.013	0.162 ± 0.028
DS	$\mathbf{0.120 \pm 0.007}$	0.307 ± 0.014	0.157 ± 0.011
NB	0.122 ± 0.007	$\mathbf{0.306 \pm 0.017}$	$\mathbf{0.151 \pm 0.014}$
JB	0.121 ± 0.006	0.307 ± 0.016	0.152 ± 0.014
NL	0.121 ± 0.016	0.308 ± 0.013	0.153 ± 0.011
JL	0.123 ± 0.014	$\mathbf{0.306 \pm 0.012}$	0.153 ± 0.013
A	0.328 ± 0.028	0.325 ± 0.034	0.324 ± 0.073
B	0.122 ± 0.017	$\mathbf{0.306 \pm 0.039}$	0.152 ± 0.025
C	0.121 ± 0.020	0.311 ± 0.041	0.154 ± 0.027
D	0.135 ± 0.014	0.310 ± 0.045	0.168 ± 0.030
E	0.123 ± 0.017	0.309 ± 0.045	0.153 ± 0.027

Table 4.5.: Bone Mineral Density results.

4.11. Summary

In this chapter we have presented six new algorithms for computing conditional quantile estimates from RKHS embeddings output by kernel Bayes' rule. These are generative quantile algorithms, based on estimate the conditional cumulative distribution from the embedding. We demonstrated two methods to derive these cumulative estimates. The first method utilised the reproducing property to directly estimate the cumulative through embedding indicator functions into the RKHS. The second method generated posterior density estimates using the MKBR algorithm described in chapter three, and integrated to determine the CDF.

We showed that the direct method may produce an estimate that is only approximately a CDF, in that it may not be strictly non-decreasing. The posterior-based methods will always produce a valid CDF.

We then constructed quantile estimators from these CDF estimators using simple numerical root-finding, and considered training on both the pinball loss function, and the negative log-probability. We then evaluated the resulting algorithms using a set of well-known machine learning datasets.

Our pre-image quantile estimators in particular were comparable to, or better than, the state-of-the-art algorithms in the literature. These algorithms can be trained either using

pinball loss, for a slight increase in performance, or with negative log-probability, to guarantee the non-crossing constraint. Additionally, all our algorithms explicitly represent multi-modal distributions, and can model datasets with strong heteroskedasticty.

Chapter 5

Scalable Multi-Modal Regression

Recent technological advancements have caused an explosion of electronic data recording many aspects of our lives and the world around us. In this chapter we present an extension to our multi-modal, non-parametric inference algorithm described in chapter three to better scale to these larger datasets.

5.1. Motivation

Mass electronic communication, ubiquitous inertial and global positioning sensors and continuous high-frequency logging of financial transactions and medical data, are just some of the sources now constantly generating new data to be analysed. The ability to make predictions from these large datasets using machine learning techniques has already demonstrated significant benefits in a number of application areas. The flagship application has thus-far been global social networking and targeted advertising, but increasingly these 'big data' techniques have tackled critical problems such as understanding the underlying genetic factors of disease [76], the causes and spread of pollution, and financial market instabilities [37].

Kernel Bayes' rule (KBR) and the multi-modal extensions described in this thesis naturally lend themselves to these sort of applications; in which analytic models can be difficult to formulate and little is known about the associated probability distributions *a-priori*. The challenge to overcome is one of scalability. A naïve implementation of KBR given

n training points is $O(n^3)$ in computation time and $O(n^2)$ in storage, precluding datasets much larger than ten thousand points on current hardware. This is very far from the terabyte scale of modern big data analytic techniques, and in fact, datasets that could easily fit on a portable USB thumb-drive would still be too large to train KBR on modern hardware.

Therefore, in this chapter we focus on modifications of our multi-modal KBR algorithm (MKBR) to lower the computation and storage complexity bounds and hence enable analysis of larger datasets: at least 10,000 points, and with a linear time complexity. We propose a novel method that is $O(m^3 n)$ in computation and $O(m^3)$ in storage with $m << n$. This method constructs a reduced training set that induces a lower-dimensional reproducing kernel Hilbert space (RKHS) in which to solve the MKBR equations. We denote the method sparse multi-modal kernel Bayes' rule, or SMKBR.

5.2. Contributions

This chapter focuses on the problem of scaling MKBR inference to be able to use more than 10,000 data points, by reducing time complexity to $O(n)$ and space complexity to $O(k)$ for n training inputs (with k a constant). To address this problem, we present the following contributions:

Novel reduced set construction algorithm for large-scale multi-modal kernel Bayes' rule inference. We present the sparse MKBR algorithm (SMKBR), based on the reduced set techniques used in support vector machines [105] and large-scale Gaussian process (GP) regression [97]. For n training points, we learn a new set of $m << n$ pseudo-samples to perform MKBR inference by minimising the posterior negative log-probability (NLP) of the full training data with respect to the new representation. The resulting lower-dimensional RKHS enables efficient treatment of very large datasets with only $O(m^2)$ storage and $O(nm^3)$ time requirements. Like the MKBR, our sparse approximation also supports multi-dimensional posterior output.

Demonstration of competitive performance of SMKBR on large datasets We compare our method with a number of scalable GP implementations, and the original scalability suggestion of a low-rank Gram matrix approximation given in [34]. We demonstrate competitive performance on a number of standard machine learning datasets in the liter-

ature, particularly in the case where significant data noise exists.

Re-derivation of low-rank strategy from [34]. We explain the low-rank approximation strategy to decrease computation cost of the KBR algorithm, show the resulting KBR equations using the Woodbury identity [126], and suggest using a well-know Nyström technique for stochastically approximating the required decomposition [25].

5.3. Related Work

The work on large-scale, functional Bayesian inference has predominantly focussed on GPs, though many of these techniques are applicable to any kernel method. Like KBR, the computational complexity of naïve GP regression for n training inputs is $O(n^3)$ in time and $O(n^2)$ in storage. There are many strategies for reducing these bounds, and with a few exceptions, these strategies are motivated by either implicitly or explicitly referencing some subset $m << n$ of the training data. For a thorough overview of these and other GP approximation techniques, see [98].

One of the original strategies for reducing GP complexity was to use low-rank approximations for the Gram matrix. For a GP, Gram matrix inversion is necessary to perform inference, but this matrix is usually $n \times n$ for n training points, which becomes intractable to invert on modern hardware around the range of ten thousand points [98]. Assuming the eigenvalues of G decay rapidly, a rank m approximation of G can be obtained by considering only the first m eigenvalues of G. With the aid of the Woodbury identity, this reduces the computational cost of Gram matrix inversion to $O(nm^2)$ in time and $O(mn)$ in storage. Unfortunately, the optimal low-rank approximation to a Gram matrix requires performing an eigen-decomposition, which is itself an $O(m^3)$ operation [39]. Therefore, approximate eigen-decompositions have been utilised to increase the efficiency of this approach.

The Nyström method is one such approximate eigen-decomposition, first applied to kernel methods in [124]. This method considers a reduced set I of m training points, and approximates the kernel evaluations of the remaining points as linear combinations of those in I. The resulting optimal approximation is of the form $G = G_{XI} G_{II}^{-1} G_{IX}$, where G_{II} is the Gram matrix of the reduced set, and G_{XI} is the $n \times n$ Gram matrix of cross-terms [109].

This Nyström method is one example of a general class of subset of data (SD) methods, that all utilise a small subset of the full set of training points. SD methods are efficient compared to standard GP inference, being $O(m^2 n)$ in time and $O(mn)$ in storage. The challenge with these methods is how to select the members of the training data subset in such a way as to maximise performance of the resultant approximation.

The brute-force approach of considering all possible subsets of a fixed size rapidly becomes intractable as m grows even beyond a few dozen points. Greedy approximations were suggested in [109, 105] which choose the next best point either from all remaining points, or a random subset. These choices are optimal with respect to sum of the RKHS distances between the true kernel functions and their reduced-set representation. Other approaches to evaluating the quality of a selection include maximising the differential entropy of the posterior evaluated at prospective points [74], maximising the log-marginal likelihood of the training data [96], and maximising the information gain as measured by the KL divergence of the new posterior probability with respect to the old [103].

A close cousin to the SD methods is a Nyström method proposed in [25]. This method uses a non-uniform probability distribution to sample the columns of G based on the modulus of the diagonal elements. Probabilistic error bounds were also derived. This method is similar to the randomised SVD approximation developed in [29], but takes explicit advantage of the symmetry of the Gram matrix, and does not use the subspace projection approach [25].

The subset of regressors (SR) methods are another popular approach to scaling GP regression. Again, these methods rely on preferencing a reduced set I of $M << N$ training points. Subset of regressors utilises the fact that the mean of a Gaussian process estimate of a function can be written as

$$f^* = E\left[\sum_{i=1}^{n} \alpha_i k(x_i, \cdot)\right] \tag{5.1}$$

where the vector $\alpha \in \mathcal{N}(\mathbf{0}, G_{XX}^{-1})$ [98]. The essence of subset of regressors is that this expression can be approximated by only considering m terms in the sum. The resulting approximation is $O(mn^2)$ in time and $O(mn)$ in storage [123, 91]. Again, choosing the m points for optimal performance is intractable, and the same reduced-set selection methods described

above can be employed.

The projected process (PP) approximation is a hybrid of the SR and SD approaches. Unlike SR, PP approximations are strictly GPs, in which the likelihood calculation for points outside the subset is approximated by considering only the mean of the posterior. The resulting predictive mean is identical to the SR case, but the variance is different [101]. The projected process also scales as $O(mn^2)$ for computation and $O(mn)$ for storage.

The Bayesian committee machine (BCM) approximates full GP regression by splitting the training data up into separate partitions, and applying a GP over each of these partitions separately. The resulting predictive distribution as assumed to be the product of predictions by the individual 'committee member' GPs. Interestingly, this produces a model which gives different answers depending on the number of query points used. Selecting the partitions can be done either randomly, or by clustering in the input dimensions [100].

One difficulty with all the above reduced set selection methods is the need to optimise the hyperparameters outside these selection strategies. The discrete choice of the reduced set induces discontinuities in the computation of the marginal likelihood, making kernel hyperparameter optimisation difficult [110]. To alleviate this problem, and improve performance, [110] considers instead a reduced set *construction* method, in which pseudo-observations are used instead of real training points, and these observations are jointly optimised along with the kernel hyperparameters using gradient-based techniques.

It turns out that many of the above scaling strategies for reducing GP complexities can be re-cast as different variations of the same approach; adding a set of $m << n$ latent variables to the model that reduce the time complexity of inference. This connection unified the above models in a single latent variable GP formulation [97].

A different approach to the GP scaling problem involves spectral decomposition and approximation of the Gaussian process [75]. The covariance function is approximated by its spectral decomposition at a finite number M of frequencies, and the resulting inference problem can be solved with $O(m^2 n)$ computation and $O(mn)$ storage. These frequencies can be selected by optimising the marginal likelihood jointly with the hyperparameters.

It was observed in [119] that all reduced set construction methods can be prone to overfitting, as induced pseudo-inputs actually induce changes in the underlying covariance function, and hence create a high-dimensional hyperparameter space if jointly optimised with the kernel parameters. Also, such methods do not explicitly minimise the approxima-

tion error to the full GP, but rather only maximise the performance of the approximation with respect to the hyperparameters. To counter these issues, it is possible to determine inducing inputs and hyperpameters though a variational inference approach to maximising the lower bound of the marginal likelihood [119]. The KL-divergence between the full GP and the induced approximation is computed, and used as part of a reduced set selection strategy, in which a new point is selected, then the hyperparameters re-optimised. This approach demonstrated more resilience to over-fitting compared with [75, 110], at the expense of the much greater computational cost associated with the variational learning.

Finally, the variational strategy was recently generalised in [47]. Using stochastic variational inference, the inducing inputs were simultaneously learned with the hyperparameters, rather than selected from the training set. Stochastic gradient descent was utilised to efficient optimisation, which was only $O(m^3)$ in time and $O(m^2)$ storage.

We now proceed to present our main contribution; an algorithm for large-scale multi-modal inference using kernel Bayes' rule, based on a reduced set construction technique and our work in multi-modal pre-image recovery in chapter three. Like the stochastic variational inference approach above, we have achieved efficient computation and storage of $O(m^3)$ and $O(m^2)$ respectively.

5.4. Sparse Multi-modal Kernel Bayes' Rule

The approach for scaling KBR inference considered in [34] was to approximate the Gram matrix inversions using a low-rank strategy. With a rank m approximation, this resulted in an algorithm $O(nm^2)$ in time and $O(nm)$ in storage.

Though efficient low-rank approximations such as [25] implicitly sub-sample the training points, the RKHS elements representing the various distributions are still functions of the training set in its totality. The training set $\{x_i, y_i\}_{i=1}^n$ for example, induces RKHS elements of the form $f = \sum_{i=1}^n \alpha_i k(x_i, \cdot)$. It is for this reason that the storage complexity of the low-rank approximation is $O(nm)$, as it must store a matrix $n \times m$ where n is the dimensionality of the underlying RKHS representation (see section 5.5.1 for details).

Therefore, further reducing the computational complexity of the MKBR algorithm requires reducing the dimensionality of the basis used to represent the embedded distribu-

tions. This can be achieved by using a reduced set method; using a smaller set of training points $\{x'_i, y'_i\}_{i=1}^m$ with $m << n$. These new training points could either be selected from the full set, or could be entirely new 'pseudo-inputs'; points whose values have been learned to produce a posterior distribution similar to that would be generated from the training data, but using many fewer points.

For MKBR, the latter, reduced set construction is a more natural approach. In chapter three, the training points were used as the means of the mixture components used in MKBR posterior estimates (section 3.5.3). Taking the same approach with learned pseudo-inputs means we would also be optimising the parametrisation of the posterior probability estimate, which up until now has been fixed *a-priori*.

Therefore, this reduced set construction approach is the one we take to scale MKBR inference. Given a set of training points $\{x'_i, y'_i\}_{i=1}^m$ from a joint distribution $P(X, Y)$, a set of weighted prior samples $\{\gamma_i, u_i\}, u_i \in \mathcal{X}$ from the prior $P(X)$, an observation y and a kernel k, MKBR as implemented in chapter three produces a posterior estimate of the form

$$P(X|Y=y) = \frac{1}{c} \sum_{i=1}^{n} w_i k(x_i, X), \tag{5.2}$$

where c is volume of the kernel and $\{w_i\}$ are positive normalised weights. From the same inputs, the sparse MKBR simply replaces the full training set $\{x_i, y_i\}_{i=1}^n$ with a set of pseudo-inputs $\{x'_i, y'_i\}_{i=1}^m$, using the same inference procedure. The resulting posterior probability estimates are of the form

$$\hat{P}(X|Y=y) = \frac{1}{a} \sum_{i=1}^{m} w_i k(x'_i, X). \tag{5.3}$$

An pseudo-code outline of our reduced set construction approach, denoted sparse multi-modal KBR (SMKBR), is given in algorithm 3.

5.4.1. Computational Complexity

The matrix inversions in the KBR equations (equations 2.94 and 2.96) are only dependent on the reduced Gram matrices $(G'_{YY})_{ij} = k_X(x'_i, x'_j)$ and $(G'_{XX})_{ij} = k_Y(y'_i, y'_j)$, meaning they

Algorithm 3: The SMKBR Regressor

Input: pseudo-inputs $\{(x'_i, y'_i)\}_{i=1}^m$, weighted prior samples $\{\gamma_j, u_j\}_{j=1}^p$, regularisers ε, δ, kernels $(k_x, \theta_x), (k_y, \theta_y)$ pre-image regulariser λ, query point (observation) y
Output: posterior mixture weights $\mathbf{w}$ (equation 5.3)

Calculate pre-image matrices A, B (Eq. 3.20);
Calculate joint $\boldsymbol{\beta}$ (Eq. 2.85) ;
Calculate $R_{X|Y}$ (Eq. 2.97 or 3.38);
Embed observation $\mathbf{k}_Y[y] = \sum_j^m k_y(y'_j, y)$;
Calculate posterior $\boldsymbol{\alpha}$ (Eq.2.96);
Find mixture weights $\mathbf{w}$ (Eq. 3.19 or 3.39);

can be computed with $O(m^3)$ cost and $O(m^2)$ storage. Similarly, the pre-image methods described in sections 3.5.4 and 3.5.3 also see only the pseudo-inputs, limiting their computational complexity to $O(m^3)$. This is determined by the quadratic program solve in the pre-image method (equation 3.19), and the matrix multiplication in the normed weights method (equation 3.39). Training still requires $O(n)$ operations however, as each of the training samples must be considered at least once.

5.4.2. Example

Figure 5.1 illustrates the flexibility of our approach. In this figure, an MKBR estimate trained on a 0.5% random subset of the full input set is contrasted with an SMKBR estimate with the same number of pseudo-observations. Immediately noticeable is that the training points in the optimised case have moved such that they attempt to cover much more of the function structure. As a result, the learned kernel bandwidths can be much lower, and hence the width of the posterior distributions smaller. The main challenge with this method is to extend this reasoning to many more data points than the case illustrated contains.

5.4.3. Parameter Learning

With the introduction of the pseudo-inputs, we have dramatically increased the number of parameters that need to be learned relative to the MKBR. For n_x-dimensional states, and n_y-dimensional states and m pseudo-inputs, at least $m(n_y + n_x)$ parameters for the

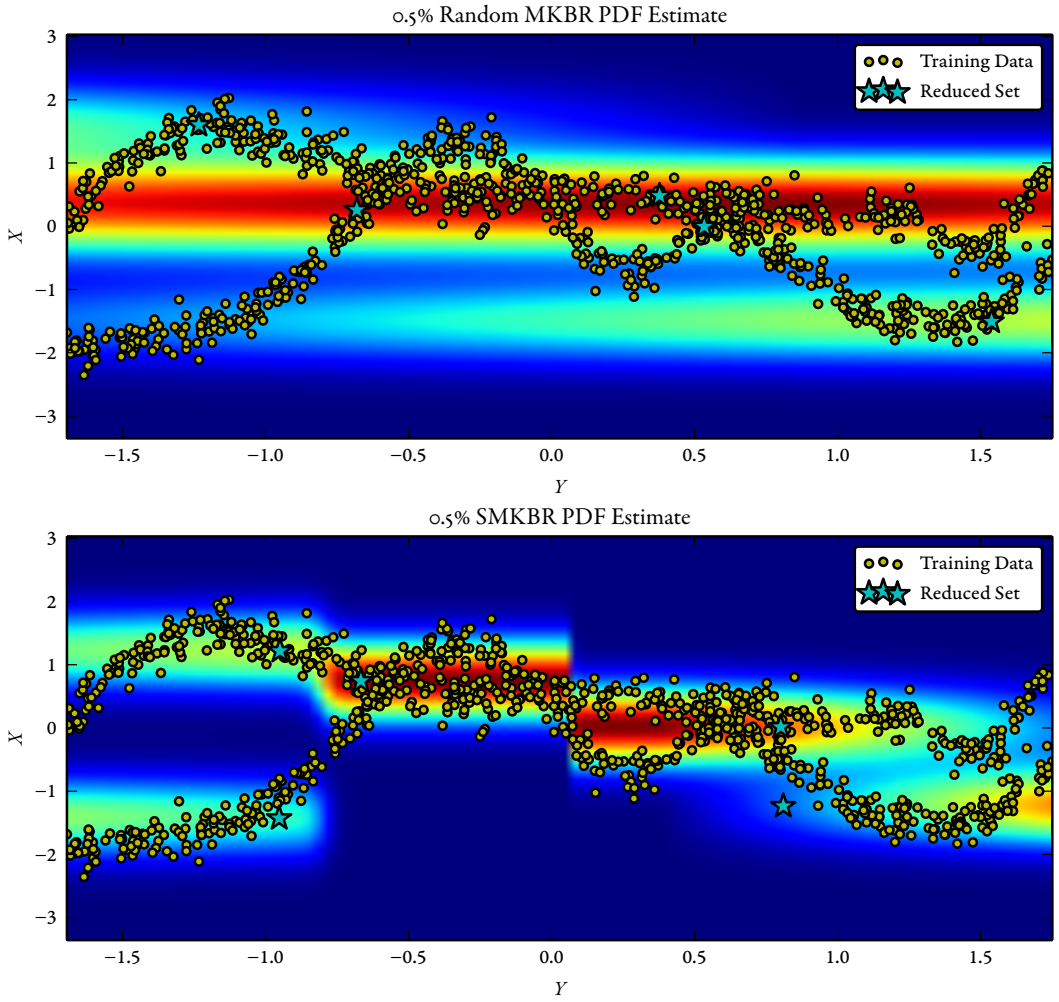


Figure 5.1.: (Top) MKBR posterior utilising a randomly-selected reduced set of 0.5% of the training data (5 points).
(Bottom) the SMKBR algorithm with 5 pseudo-inputs. The mixture widths learned in this case are much smaller, owing to the smart placement of the pseudo-inputs. The discontinuity evident at $Y = 0.0$ is a numerical artefact owing to the low number of points in the basis sets, and is caused by a large regulariser value δ being learned.

pseudo-inputs, on top of any kernel parameters and the two regularisation terms in KBR.

We can employ the same cost function as used to train the parameters of the MKBR in chapter three, to jointly train the pseudo-inputs as well. This function is, as in section 3.6, the NLP of the testing states when conditioned on the training states. Let $\{(x_i, y_i)\}_{i=1}^n$ be samples from the joint $P(X, Y)$, and let $\hat{P}(X|Y=y_i; \theta)$ be the posterior estimate produced by the SMKBR algorithm when conditioned on a training observation y_i and with unknown parameters θ. The NLP cost function is then

$$C(\theta) = -\sum_{i=1}^{n} \log \hat{P}(x_i|Y=y_i; \theta). \tag{5.4}$$

In chapter three, we used k-fold cross-validation to prevent over-fitting of the data with this cost function. However, if the number of pseudo-inputs is much smaller than the number of training points, the danger of over-fitting is reduced. Additionally, unlike chapter three, the mixture components of the posterior estimate are not centred on the training points (but rather at the pseudo-inputs). This further reduces the danger of over-fitting, because the test points are not being evaluated at the mode of each mixture component.

Initialisation

Initialisation of the parameters at the beginning of the training is critical in such a large search space, especially with a non-convex cost function such as equation 5.4. Following [47], we initialise the $(x', y')_{i=1}^m$ pseudo-inputs using the k-means algorithm run on the training data [77]. For the kernel parameters and the regularisation terms, we use the result of an initial optimisation that holds the pseudo-inputs fixed.

Optimisers

For the initial optimisation of the kernel parameters and the KBR regularisers, we use a global optimiser called controlled random search (CRS) with local mutation [57], which initialises a large population of points in the search space and then perturbs these points using simplex-based local update rule. Based on our experiments, CRS is more biased toward global rather than local search and so performs better on larger datasets compared

with the multi-level single linkage (MLSL) [59] algorithm used in chapter three.

The joint optimisation of the parameters and the pseudo-inputs is performed using the local optimisation strategy denoted constrained optimisation by linear approximations (COBYLA) [93]. This is a derivative-free algorithm that constructs successive linear approximations of the cost function to perform gradient descent. As in chapter three, both the CRS and COBYLA implementation are part of the 'NLOpt' software package [54].

The use of a derivative-free algorithm in this context is less than ideal. For a large parameter space (as is the case with a large set of m pseudo-inputs), an efficient analytic expression could significantly speed up training. It is also worth noting that equation 5.4 is the sum of independent cost terms, and is therefore amenable to stochastic gradient descent [15]. This would allow potentially more efficient computation of an approximate derivative to further speed up training for very large training sets. Implementing these features is a subject of future work.

5.5. Other MKBR Scaling Methods

In addition to the SMKBR algorithm forming the main contribution of this chapter, we also present two simple scaling strategies for MKBR against which it is insightful to compare. The first is simply random reduced set selection, in which we decimate the training set randomly to contain only m points. The second is the low-rank strategy mentioned in [34], with the addition of a Nyström-based stochastic Gram matrix approximation.

Random Subset of Data

Random reduced set selection for MKBR simply requires taking a subset $\{x_i, y_i\}_{i\in I}$, $|I| = m$ of m pairs of x_i, y_i points from the original training set $\{x_i, y_i\}_{i=1}^{n}$. The pairs of points are sampled randomly without replacement. The reduced training set is then used to train the MKBR. Assuming the reduced set is much smaller than the original, (i.e. $m << n$), then cross-validation in the learning becomes unnecessary, and we can simply use the decimated samples $\{x_i\}_{i\in(1..n)\setminus I}$, with the same NLP cost function as the SMKBR, equation 5.4.

This random subset of data method forms a baseline from which we can compare our algorithm. It allows us to measure the value of optimising the pseudo-inputs over what is

the simplest reduced set method.

5.5.1. Low-Rank Gram Matrix Approximation

Gram matrix inversion is the dominating operation that defines the time complexity of MKBR. Using a low-rank approximation to the Gram matrix provides an efficiency gain in this inversion, both in time and space complexity. In this section we outline the method suggested in [34] for low-rank approximation of KBR, which of course also applies to MKBR.

Naïvely, Gram matrix inversion is an $O(n^3)$ operation with n training points. However, it may be the case that the eigenvalues of a Gram matrix G decrease rapidly, implying it is possible to approximate the $n \times n$ Gram matrix as the product of a tall $n \times m$ matrix and a long $m \times n$ matrix, where $m << n$. In this case, the resulting matrix inversion is $O(mn^2)$ in storage and $O(mn)$ in computation.

The two matrix inversions required in the KBR are the computation of the joint embedding in equation 2.94, and the computation of the posterior operator $R_{X|Y}$ in equation 2.97. The first computes the weights α of the joint embedding $\mu[X, Y] = \mathbf{K}_{XY}^T[\cdot]\boldsymbol{\beta}$ for the training points $\{x_i, y_i\}_{i=1}^n$ as

$$\boldsymbol{\beta} = (G_{XX} + \varepsilon I)^{-1} G_{XU}\boldsymbol{\alpha}. \tag{5.5}$$

Approximating G by a rank m matrix requires finding the $m \times n$ matrices U and V^T and the $m \times m$ matrix B such that $\hat{G} \approx G = UBV$. Actually computing the low-rank factorisation can be achieved in a number of ways. For small matrices, computing the singular value decomposition (SVD) and throwing away the smallest $n - m$ eigenvalues is tractable, but for larger matrices, approximations are required. One efficient approach is to use randomised SVD algorithms [43] that compute an approximate SVD through random projections of the matrix into lower-dimensional sub-spaces. This means we need never actually store the full $n \times n$ matrix.

For our work we implemented a popular algorithm specifically designed for Gram matrices, based on these randomised matrix decompositions [25]. For an $n \times n$ Gram matrix, this algorithm produces a rank m approximation of the form $\hat{G} = CW^{-1}C^T$, where C is a $n \times m$ matrix, and W is an $m \times m$ invertible matrix. The matrix C is constructed by randomly

drawing columns from G (with replacement) according to a data-dependent probability distribution. The matrix C is the intersection of these columns with the corresponding rows of G.

With this low-rank approximation of G, the Woodbury identity describes how to compute the sum of G and a full-rank matrix A with only two matrix inversions of size $m \times m$, provided A^{-1} is known [126].

Definition 5.5.1. Given a full-rank matrix A, and a rank N matrix UBV, where U and V^T are $N \times M$ and W is $M \times M$, then the Woodbury identity asserts that

$$(A + UBV)^{-1} = A^{-1} - A^{-1}U\left(B^{-1} + VA^{-1}U\right)^{-1} VA^{-1}. \tag{5.6}$$

In the case of equation 5.5, the full-rank matrix A is diagonal, making its inversion trivial. There are then only two $M \times M$ matrix inversions to perform. Substituting the decomposition of G into this equation and expanding with the Woodbury identity with $U = C$, $V = C^T$, $B = W$ and $A = \varepsilon I$ yields

$$\boldsymbol{\beta} = (\hat{G}_{XX} + \varepsilon I)^{-1} G_{XU}\boldsymbol{\alpha} \tag{5.7}$$

$$= \frac{1}{\varepsilon} G_{XU}\boldsymbol{\gamma} - \frac{1}{\varepsilon^2} C(W + \frac{1}{\varepsilon} C^T C)^{-1} C^T G_{XU}\boldsymbol{\gamma}. \tag{5.8}$$

For the details of computing C and W, see [25]. For better numerical stability, direct inversion can be replaced with linear solves. In this case, finding $\boldsymbol{\beta}$ amounts to solving

$$C(W + \frac{1}{\varepsilon} C^T C)\mathbf{x} = C^T G_{XU}\boldsymbol{\gamma} \tag{5.9}$$

and then substituting x giving

$$\boldsymbol{\beta} = \frac{1}{\varepsilon} G_{XU}\boldsymbol{\gamma} - \frac{1}{\varepsilon^2}\mathbf{x}. \tag{5.10}$$

The second Gram matrix inversion to compute the KBR posterior is part of the equation

for the conditioning operator $R_{X|Y}$. From equation 2.97:

$$R_{X|Y} = D(\beta)G_{YY}\left((D(\beta)G_{YY})^2 + \delta I\right)^{-1} D(\beta), \tag{5.11}$$

where $D(\beta) = \text{Diag}(\beta)$. The different kind of regularisation here requires a slightly different form of approximation. If $G_{YY} = CW^{-1}C^T$ then applying the Woodbury identity with $U = D(\beta)CW^{-1}C^TC, B = W^{-1}$ and $V = C^T$ yields

$$R_{X|Y} = \frac{1}{\delta}D(\beta)G_{YY}D(\beta) \tag{5.12}$$

$$- \frac{1}{\delta^2}D(\beta)CW^{-1}C^TC(W + \frac{1}{\delta}C^TD(\beta)CW^{-1}C^TC)^{-1}C^TD(\beta). \tag{5.13}$$

The associated linear equations are

$$W\mathbf{x} = C^TC \tag{5.14}$$

$$(W + \frac{1}{\delta}C^TD(\beta)C\mathbf{x})\mathbf{y} = C^TD(\beta) \tag{5.15}$$

$$R_{X|Y} = \frac{1}{\delta}D(\beta)G_{YY}D(\beta) - \frac{1}{\delta^2}D(\beta)C\mathbf{xy}, \tag{5.16}$$

which require solving for $\mathbf{x}$, then $\mathbf{y}$, then substituting into the expression for $R_{X|Y}$.

Using the normed weights technique in section 3.5.4 to derive a pre-image estimate, the computation of $R_{X|Y}$ can be greatly simplified. In this case,

$$R_{X|Y} = \left(D(\hat{\beta})G_{YY} + \delta I\right)^{-1} D(\hat{\beta}) \tag{5.17}$$

and so by the Woodbury identity,

$$= \frac{1}{\delta}D(\hat{\beta}) - \frac{1}{\delta^2}D(\hat{\beta})C(W + \frac{1}{\delta}C^TC)^{-1}C^TD(\beta). \tag{5.18}$$

When n is small, it makes sense to compute $R_{X|Y}$ once during the training phase, and then to condition on a particular query point y_s simply by multiplying; $\mu[X|Y = y_s] = R_{X|Y}\mathbf{k}_Y[y_s]$. Unfortunately, this requires storing the $n \times n$ matrix $R_{X|Y}$. Therefore, we can

instead solve the linear equations

$$D(\beta)G_{YY}\left((D(\beta)G_{YY})^2 + \delta I\right)\mu[X|Y = y_s] = D(\beta)\mathbf{k}_Y[y_s] \tag{5.19}$$

for every query point, reducing the storage cost to $O(n \times m)$ at the cost of increasing the time complexity by a factor equal to the number of query points.

A comparison of the low-rank MKBR to the full-rank algorithm using a toy data set is illustrated in figure 5.2. Note that although the low-rank approximation provides a cruder estimate of the KBR operators, the underlying RKHS for representing the embedding remains the same. The low-rank approximation with a rank of $0.005n$ shows some loss of detail, particularly in the sections with multi-modality when compared to the full MKBR solution. Both methods utilised normed weight pre-images and a Gaussian kernel.

Rank-m approximations of the Gram matrices still require $O(mn)$ storage and $O(mn^2)$ computation however. The main motivation for SMKBR was to reduce these complexities further, as they may still be prohibitive for very large datasets.

5.6. Experiments

In this section we evaluate the performance of SMKBR with a set of regression experiments. We compare our algorithm to a number of scalable GP approximations from the literature, as well as the MKBR with random reduced set selection, and with low-rank Gram matrix approximation. The experiments are a toy dataset consisting of a bi-modal pair of trigonometric functions, and two well-known datasets from the literature; Abalone, which aims to predict the age of abalone in Bass Strait based on a set of physical measurements, and Kin40k, which aims to predict the position of a simulated robot arm based on a set of joint rotation parameters.

5.6.1. Datasets

Toy dataset

Our first experiment is a demonstration on a toy dataset of multiple trigonometric functions. At any point x the distribution is the sum of two functions with Gaussian noise,

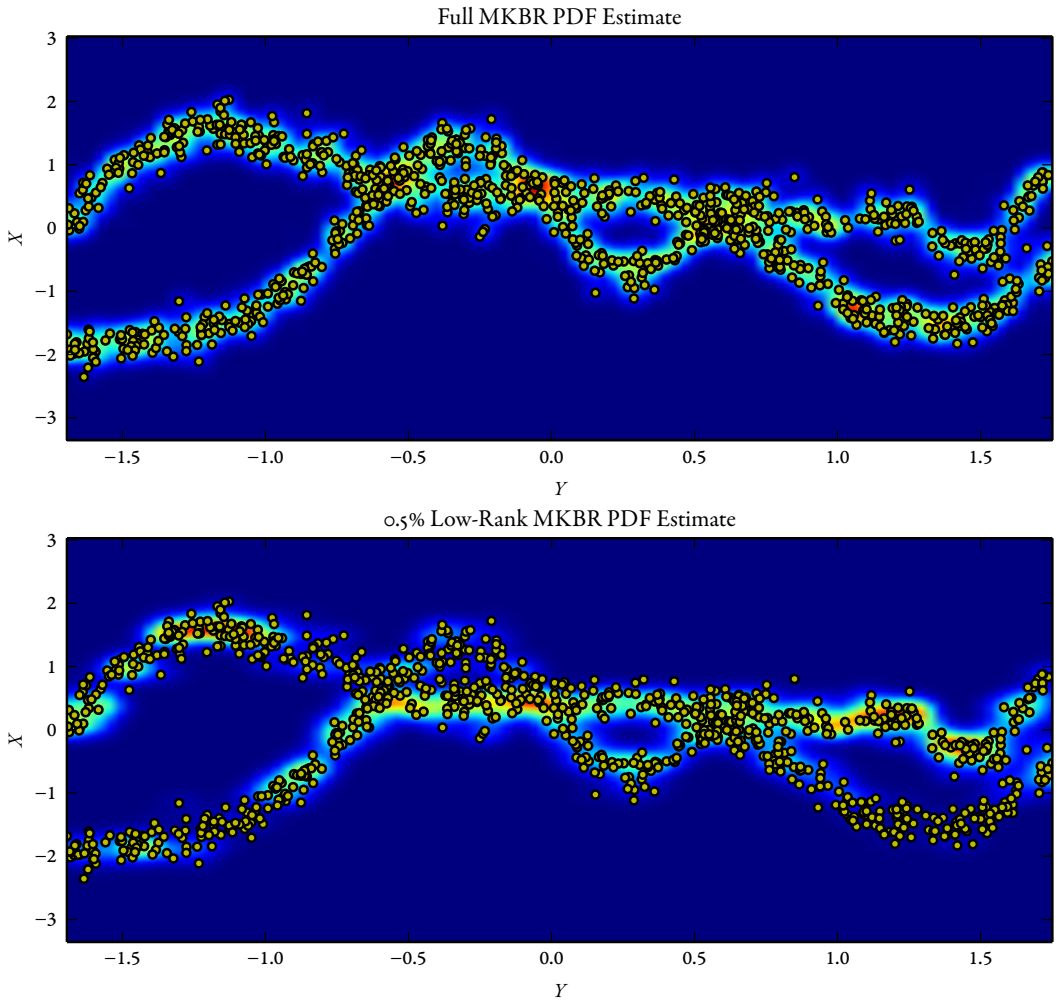


Figure 5.2.: (Top) Full-rank MKBR posterior estimate.
(Bottom) 0.5%-rank MKBR posterior estimates. Note that even with this severe approximation, the algorithm still performs well. However, a loss of detail is clearly visible, especially in the bi-modal sections of the distributions.

$P(x) = 0.5m_1(x) + 0.5m_2(x) + 2\varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma)$ and

$$m_1 = \sin(3/2\pi x) + 5(0.5 - x)\sin(3\pi x), \tag{5.20}$$

$$m_2 = -3\cos(2x)\cos(6\pi x^2), \tag{5.21}$$

and noise $\sigma = 0.3$. We consider $x \in (0, 1)$, and draw 1000 points for training and 1000 for testing. These points are then scaled to have mean 0 and unit variance. The resultant dataset, which is bi-modal, is also used in the examples given in figures 5.3 and 5.2.

Abalone

The Abalone data are measurements of nine physical attributes of abalone specimens taken from coastal waters off Tasmania, Australia [83]. There are nine attributes in the dataset: sex, length, height, total weight, shucked weight (shell removed), 'viscera weight', shell weight, and rings. Measuring the rings of an abalone is an accurate way to determine the age of the creature, but is difficult and time consuming. Therefore, the goal of the inference problem is to use the eight simpler-to-obtain measurements to predict the rings variable.

There are 4177 abalone specimens in the data, which is randomly split into 3133 training values and 1044 test values. All variables are treated as being continuous. The dataset is available from the UCI machine learning repository.[1]

Kin-40k

The Kin-40k dataset contains simulated measurements generated from an eight link all-revolute robot arm, similar to a Puma 560 model [36]. There are eight measurements of angular displacement for each of the joints, as well as a measurement of the distance between the end-effector and a fixed target. The aim of the experiment is to predict the distance to target given the joint angles. This variation of the Kin-40k dataset was simulated with high non-linearity, but low noise on the observations.

There are 40,000 data points in total, which for each run of an experiment are randomly divided into 10,000 training points and 30,000 testing points. The dataset is available

[1] http://archive.ics.uci.edu/ml/

online.[2]

5.6.2. Comparisons

The following algorithms from this thesis and the literature were examined in our experiments. The comparison results for these algorithms were reproduced from the results in [119]. The following paragraphs give a brief description of the methods, but for more detail, see the stated reference or section 5.3.

Low-Rank MKBR This is the MKBR using the Nyström-based low-rank Gram matrix approximation, described in section 5.5.1. We denote this algorithm as LR-MKBR.

Subset of Data MKBR This is the MKBR using a simple randomly-selected subset of the data, as described in section 5.5. We denote this algorithm as SD-MKBR.

***k*-Means MKBR** This is the MKBR using a reduced set construction which was computed using *k*-means on the training data, without further optimisation. We denote this algorithm as KM-MKBR.

Sparse Pseudo-Input GP This is the reduced set construction technique described in [110], in which a set of pseudo-inputs to the GP are jointly optimised with the kernel hyperparameters. We denote this algorithm as SP-GP.

Variational Lower Bound GP This is the algorithm described in [119] in which a set of inducing variables that simplify GP inference are learned by minimising a variational lower bound between the true GP output and the approximation. We denote this algorithm as VAR-GP.

Projected Process GP This is the reduced set algorithm closely related to subset of regressors, in which likelihood calculation for points outside the reduced set are approximated through the posterior mean [101]. We denote this algorithm as PP-GP.

[2] http://ida.first.fraunhofer.de/~anton/

Full GP The abalone dataset was small enough that a full GP can also can be run for comparison [98]. For more details on GPs, see section 3.7.1.

5.6.3. Experimental Setup

For every dataset, the data were rescaled such that the training points $\{y_i\}$ had zero mean and unit variance on every dimension. The $\{x_i\}$ training points were centred to have zero mean. Each experiment was repeated five times using randomly selected training and testing sets. For the Abalone and Kin-40k datasets, the size m of the reduced set was evaluated from 16, in powers of two up to 512. For the toy dataset, reduced set sizes from 1 to 20 were evaluated.

To conform with similar work in the literature, the experiments were assessed using a slight variation on the NLP measure utilised in chapter three: the standardised negative log-probability (SNLP) [98]. This measure is the difference between the negative log-posterior probability of the algorithm, and the negative log-probability of a single Gaussian with mean and variance determined empirically from the data. Therefore, a score of 0 corresponds to the most basic Gaussian regression, and a more negative score indicates better performance. Let the training data $\{x_i, y_i\}$ have mean μ and covariance Σ. Then given a testing set $\{x_i^*, y_i^*\}_{i=1}^T$, the SNLP for a predictive distribution $P(X|y_i^*)$ is

$$\text{SNLP} = -\frac{1}{T}\left[\log\left(P(x_i^*|y_i^*)\right) - \log\left(\mathcal{N}(x_i^*; \mu, \Sigma)\right)\right]. \tag{5.22}$$

For all the KBR-based methods, a Gaussian kernel was used with a diagonal covariance matrix, and a single parameter for each dimension of x and y. As in chapter four, the training data were whitened to remove covariance between dimensions. For very large data sets, it is probably not feasible to perform a pre-image optimisation at every evaluation of the cost function. Therefore, we utilised the normed-weights method described in section 3.5.4 for all experiments.

5.6.4. Results

Toy Dataset

The results of the toy dataset are illustrated in figure 5.3, and a full table of results is given in table B.2. The best performing algorithm after the full MKBR is the low-rank MKBR. This is unsurprising, considering the full dimensionality of the Hilbert space is retained in this method, at the cost of additional storage and computation. Interestingly, the performance of this method appears to be roughly independent of the size of the reduced set. This is likely due to the intrinsic rank of the full Gram matrix being quite low. The difference between the SMKBR and the KM-KBR illustrates the benefit of training the pseudo-inputs over simply using their initialised values from the k-means algorithm. Finally, the SD-MKBR method performs the worst. Given that its reduced set is randomly selected, there are likely significant areas without coverage by a training point for small data fractions.

Abalone Dataset

The Abalone results are plotted in figure 5.4, and tabulated in table B.3. Here we see the full MKBR dramatically outperforms all other methods including the full GP. The abalone dataset is known to be quite noisy, and without making the Gaussianity assumption the MKBR is able to model a much richer class of conditional distributions. The SMKBR also demonstrated good convergence toward the full MKBR result as the number of points increased.

Kin-40K Dataset

The Kin-40k results are plotted in figure 5.5, and tabulated in table B.4. In this case the performance of the SMKBR was less impressive than the majority of the other methods, especially at larger set sizes. There are a few possible reasons for this. The first is that as Kin-40k is a low noise dataset with essentially deterministic behaviour, high performance in this dataset speaks more to the quality of the latent function prediction, rather than the posterior distribution. Because SMKBR has significantly relaxed the assumptions on the posterior relative to a GP, it would be expected to perform worse when the GP assumptions

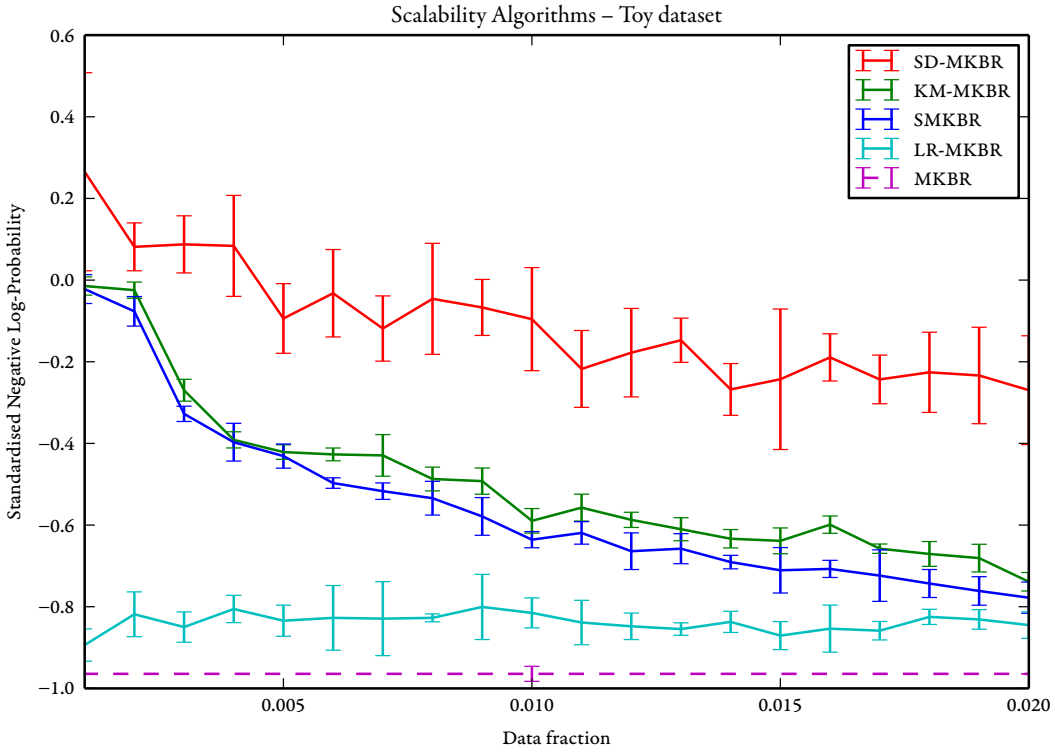


Figure 5.3.: Results from the sinusoidal toy regression problem. GP results were omitted due to the string multi-modality of the problem.

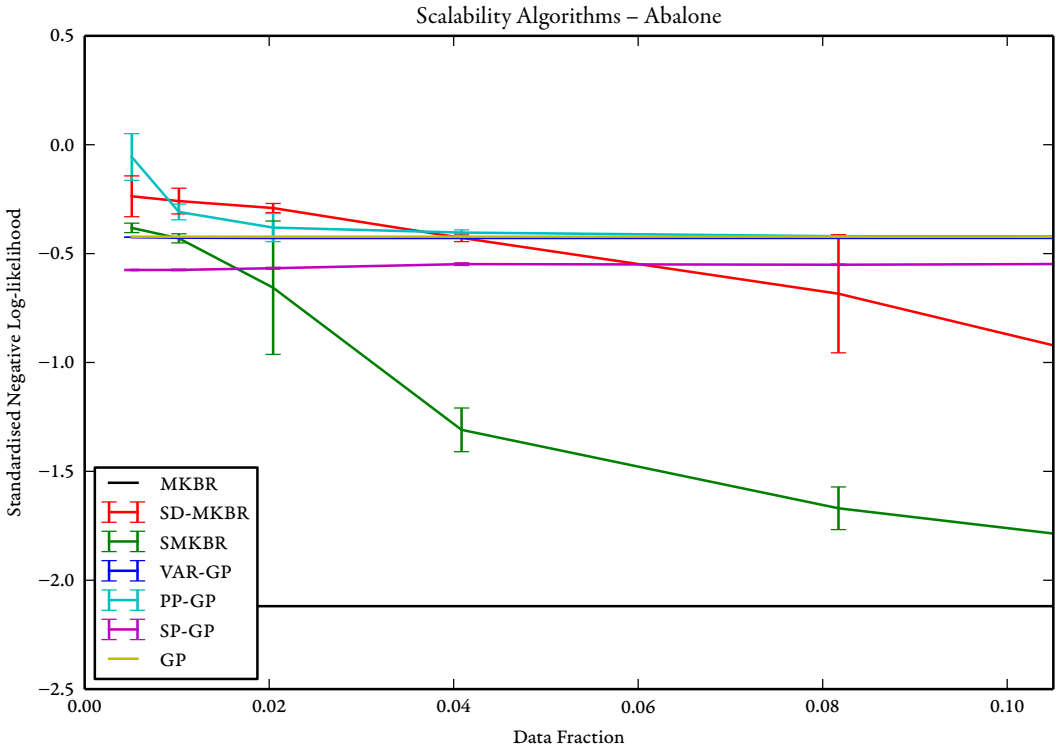


Figure 5.4.: Results from the Abalone dataset. Note that the SPGP likely performs better than the full GP despite being an approximation, because of its ability to model heteroscedastic behaviour.

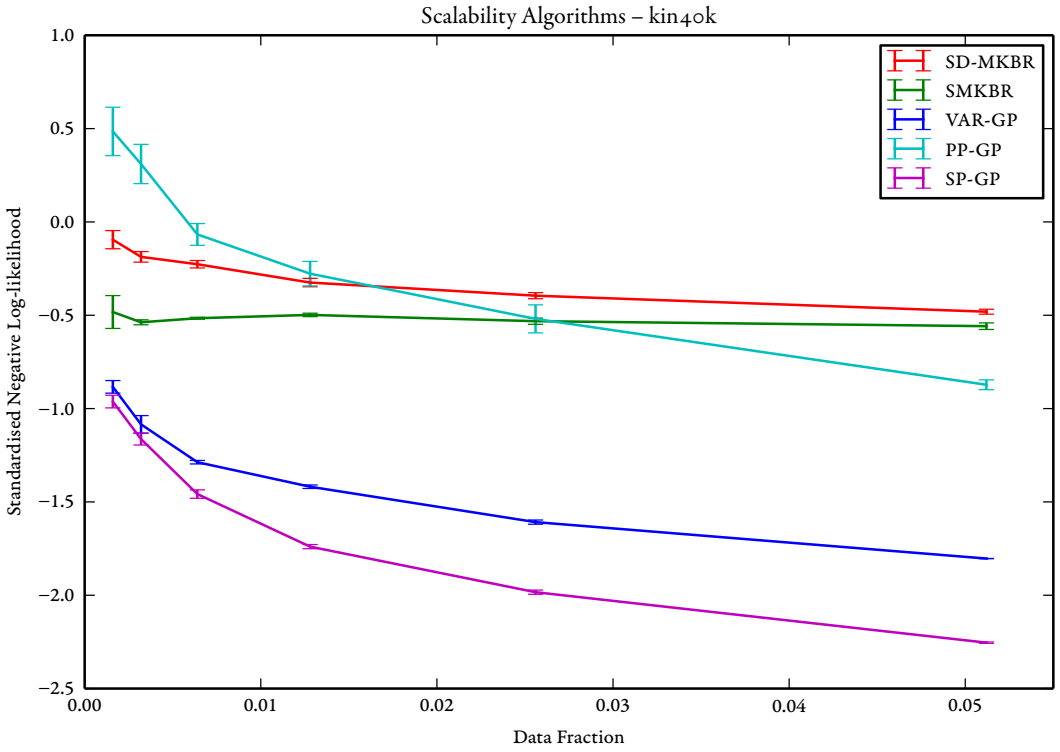


Figure 5.5.: Results from the Kin-40k dataset

are valid (as they appear to be in this case). Another reason is that the larger dataset made training more difficult, and our implementation was somewhat limited in training speed by the use of numerical derivatives for the training scheme.

5.7. Summary

In this chapter we have presented a new algorithm for efficiently scaling multi-modal kernel Bayesian inference to large datasets. We have utilised a reduced set construction technique to lower the time complexity of MKBR from $O(n^3)$ to $O(m^3)$, and the storage from $O(n^2)$ to $O(m^2)$, where m is the number of pseudo-input parameters. Rather than using the n training inputs in the KBR directly, our algorithm instead uses a set of $m << n$ pseudo-inputs. To train the algorithm, we perform a joint optimisation of the pseudo-inputs, kernel parameters and regularisation terms, using the negative log-probability of

the training data as a cost function. We named the resultant algorithm sparse multi-modal KBR or SMKBR. We have also re-derived the low-rank approximation to KBR suggested in [34], using a Nyström-based method for stochastic approximation of the Gram matrices.

We demonstrated the effectiveness of the SMKBR algorithm on a set of experiments; a simulated bi-modal regression problem, and the Abalone and Kin-40k datasets from the literature. The abalone experiment attempted to use physical measurements of abalone specimens to estimate their age, whilst the Kin-40k experiment attempted to learn the position of a robot end-effector given the angular position of its eight rotational joints. Our algorithm performed competitively with state-of-the-art scalable Gaussian process techniques from the literature. This was particularly true in the Abalone dataset, where SMKBR outperformed all other algorithms including sparse pseudo-input GPs [110] and variational lower bound GPs [119].

Chapter 6

Conclusions

This thesis developed multi-modal, scalable, non-parametric inference using kernel Bayes' rule (KBR). Inference is the foundation stone of humanity's system of knowledge, and when quantified through Bayesian probability, it is a fundamental philosophical tool for how science determines what is 'true'. Fortunately for the richness and diversity of the world around us, many physical, sociological and environmental phenomena display complex, non-linear behaviour which eludes analytic description. Unfortunately for statisticians, these properties make systems difficult to model with traditional parametric techniques.

Non-parametric methods such as Gaussian process (GP) regression are able to learn the behaviour of complex phenomena directly from observations, without the need for *a-priori*. However, GPs are limited to modelling normally-distributed uncertainty in their predictive models.

Kernel Bayes' rule is another type of non-parametric inference system that relaxes the Gaussian assumptions inherent in GP regression. It is able to learn prior and likelihood functions from a set of sample observations without making assumptions about the nature of the underlying distribution, whilst still being able to represent non-linear phenomena in multiple dimensions. It utilises high-dimensional function spaces to embed representations of the prior and likelihood distribution, and perform a Bayes update inside the function space. The result of the algorithm is an element of the space corresponding to an embedded posterior distribution. Expectations or a maximum *a-posteriori* estimate can be

obtained from this embedding.

Though these estimates are useful by themselves, understanding the full posterior density of an inference problem is important in many applications. This is especially true when that distribution contains multiple modes. For risk-based decision making, accurate quantile estimates allow the risk of particular outcomes to be bounded and so are an important product of inference. The ability to recover these estimates from KBR would significantly enhance its utility, as would reducing the computational complexity of the algorithm for scaling to larger datasets.

The objectives of this thesis were three-fold: to extend kernel Bayes' rule to recover an estimate of the full conditional posterior distribution, to estimate conditional quantiles to support risk-based descision making, and to increase the numerical efficiency of these algorithms allowing them to scale to larger datasets.

Achieving these objectives would create a system of non-parametric inference that made no assumptions about the prior, likelihood or posterior distribution and could represent multi-modal posteriors arising from complex systems. It would also be useful for understanding larger quantities of data, and using this understanding to make explicit decisions based on a rigorous process of risk quantification.

6.1. Summary of Contributions

To achieve our objectives, we developed the multi-modal kernel Bayes rule algorithm (MKBR), successfully applied it to conditional quantile estimation, and then developed a scalable extension of the algorithm that allows for inference over larger datasets. Specifically we presented the following contributions to achieve our objective:

Multi-modal Regression and Filtering

A novel mixture distribution pre-image extension to KBR Posterior estimates from KBR take the form of elements of a reproducing kernel Hilbert space (RKHS). Recovering an estimate of the distribution requires solving the pre-image problem; the reverse mapping between the RKHS and the space of probability distributions. We presented a novel approach to this difficult problem which was able to recover an estimate of the

full posterior density from KBR, by embedding a candidate mixture distribution into the RKHS space, then optimising the mixture weights to match the candidate and posterior embeddings. The method is flexible, in that arbitrary mixture components can be chosen, and efficient, requiring only the solution of a convex quadratic program to recover the density estimate. We named this algorithm multi-modal kernel Bayes' rule, or MKBR.

A new training scheme for automatic parameter estimation The MKBR algorithm required a training scheme for optimising the free parameters that properly accounted for the multi-modality of the resultant posterior. As such, standard functions like the mean-squared error were unsuitable. We presented a training technique that optimised the negative log-probability of the training data using k-fold cross validation to prevent over-fitting.

Application to difficult motion estimation problems in robotics We applied MKBR to a number of difficult recursive Bayesian filtering and regression problems in the domain of robotics. We illustrated that MKBR is able to represent high-dimensional multi-modal distributions and learn latent behaviour from data in a filtering experiment that tracked a simulated particle choosing random forks in a fixed path. We showed that a complex underlying motion model could be learned from noisy inertial measurements in a filtering experiment predicting the location of a slot car moving around an 11 metre track. Finally, we significantly extended a robotic path planning algorithm that used probabilistic velocity fields generated from tracking pedestrians. Unlike the GP-based standard method, MKBR could properly model the multi-modal velocity distribution caused by pedestrians moving in opposite directions down the same hallway. This resulted in a much more flexible system for robot motion planning.

Conditional Quantile Estimation

Novel methods to recover cumulative distribution estimates directly from KBR posterior embeddings We presented two different techniques for directly recovering estimates of the conditional cumulative distribution from a KBR embedding, without the need to first recover a PDF estimate through solution of the pre-image problem. These estimates used embedded indicator functions which took advantage of the reproducing property of RKHSs to recover cumulative estimates with a simple inner product.

Extension of multi-modal KBR to generate cumulative and quantile estimates As the output of MKBR is a posterior density estimate, recovering the cumulative distribution and hence the quantiles is simply a matter of integration and numerical root-finding. We demonstrated automatic parameter estimation of this MKBR variant through maximising the negative log-probability of the posterior density, or if a particular quantile is required, through optimisation of the pinball loss. The former guarantees that different quantile estimates will not cross, whilst the latter directly optimises the accuracy of the quantile estimate.

Competitive performance in conditional quantile regression against state-of-the-art techniques We demonstrated the various conditional quantile estimation algorithms presented by testing them on a variety of standard datasets from the literature. We compared against a number of state-of-the art algorithms, including heteroskedastic GP quantile estimation, and quantile support vector machines. Our methods, particularly those based on the MKBR posterior density, demonstrated competitive and in some cases superior performance. We showed a trade-off between the pinball loss training method, and the log-probability method. The former demonstrated slightly better performance overall, at the expense of guaranteeing the estimated quantiles do not cross.

Scalable Multi-modal Regression

Novel reduced-set construction algorithm for efficient multi-modal inference on large datasets We presented scalable multi-modal KBR (SMKBR), an algorithm that applies the MKBR to a set of pseudo-inputs then optimises those inputs to maximise performance. The result is that the underlying representations of distributions scale not with the training data but with the number m of pseudo-inputs. For n data points, this reduces the complexity of KBR from $O(n^3)$ in time and $O(n^2)$ in space to $O(m^3)$ in time and $O(m^2)$ in space, where $m << n$. The pseudo-inputs are jointly optimised with the algorithms' other parameters by minimising the negative log-probability of the training data. The resulting algorithm can be applied to much larger datasets than the standard MKBR.

Competitive performance in large regression problems against state-of-the-art scalable GP techniques We applied SMKBR to standard large datasets used in the scalable GP

community. Our algorithm demonstrated superior performance in the noisy Abalone dataset, even when compared to state of the art techniques such as sparse pseudo-input GPs, and variational GPs. The low-noise Kin-40k dataset proved more challenging, but performance was still comparable with techniques such as projected-process GPs.

6.2. Future Work

The gamut of future work arising from this thesis spans many interesting avenues of pursuit. This includes formal verification of some of the empirically demonstrated results, increases is the expressiveness and efficiency of our algorithms, and novel application areas. We briefly outline some of these possibilities below.

Convergence proofs Producing proofs of convergence for the MKBR posterior and quantile estimators is an important piece of future work. However, this relies on as-yet-unproven convergence results of KBR itself. It would also be interesting to explore the requirements on the posterior mixture representation given a particular true posterior. Another important question is the link between KBR and GP regression. This may help uncover the physical interpretation of the matrix regularisation parameters in KBR.

New kernels Extending MKBR to use more sophisticated kernel functions, especially non-stationary kernels, would potentially increase the performance of the algorithm when known information about the underlying system such as changing length-scales could be encoded explicitly.

Performance away from data Like all non-parametric methods, KBR and MKBR perform poorly when queried far from training data points. However, the behaviour of these algorithms in this domain is currently not well understood. Contrast this with GP regression, in which the choice of kernel is able to determine this behaviour. An important piece of future work is trying to characterise how the choice of kernel, regulariser, and mixture distribution impact estimates far from data.

Entropy estimation Efficient estimation of the entropy of an RKHS embedded distribution would allow established techniques for decision making based on maximising information gain to be applied to KBR. One promising avenue to explore in this area is the maximum mean discrepancy measure, an analogue for entropy computable inside the RKHS.

Missing data The MKBR is able to perform inference when both the input and output is multi-dimensional. Supporting partial input data, in which some data-points are missing some dimensions of input, is still an open problem however. Progress in this area would broaden the applicability of MKBR to the common multi-task inference problems where different measurements are not co-located.

Scalability performance For the scalable MKBR variant, future work includes implementing analytic derivatives of the cost function to increase the efficiency of training, and testing the performance of stochastic gradient descent as an optimiser. These two changes together could create a significant performance increase and allow scaling to even larger problems.

New application areas Finally, there is interesting future work to be done in applying the work in this thesis to other application areas beyond robotics and machine learning. Areas such as environmental modelling contain sparse data from multiple sensor modalities, and often lack analytic models for the underlying phenomena. Because of potential to affect the human population, accurate characterisation of uncertainty is critical. These are all factors which suggest the work in this thesis may be able to make a future contribution to the area. Another promising domain is geophysical inversion. The unconstrained inversion of geophysical measurements taken at the surface to recover structural information deep underground is fundamentally ill-posed, creating a complex posterior distribution. Though analytic models exist for many geophysical sensors, they are often expensive enough to compute that a sample-based approach such as MKBR may be able to provide efficient approximations.

Of course, the work in this thesis constitutes only a small fraction of the possible applications of kernel Bayes' rule, which is itself only part of a larger category of non-parametric

kernel-based inference techniques. The enormity of the challenge this thesis represented to the author only underlies the sheer scale, not only of the knowledge out in the world waiting to be acquired, but also of that knowledge which has yet to be discovered. It is an exciting and humbling prospect.

Now small fowls flew screaming over the yet yawning gulf; a sullen white surf beat against its steep sides; then all collapsed; and the great shroud of the sea rolled on as it rolled five thousand years ago.

(Herman Melville, "Moby-Dick; or, The Whale")

Appendix A

Foundations of Probability Theory

Probability theory describes how to reason with incomplete information. Though the field is both wide and deep, the foundations of probability theory are simple and merely formalize a task performed by every human every day. One compelling formulation of probability theory starts with the goal of extending Aristotelian logic to a theory able to reason about degrees of belief, rather than the absolutes *true* and *false*. From this goal, some basic properties of the theory can be fixed to ensure it is compatible with human experience and intuition.

Cox first wrote down three of such statements as axioms, and then proceeded to prove that any theory of reasoning based on them would be isomorphic to probability theory [21]. This was an important result, because at the time the prevailing interpretation of probability theory was that it represented the result of large numbers of repeated trials rather than a degree of plausibility. Since then, the original set of axioms has been refined, and some of Cox's assumptions relaxed [51]. There have also been quite different axiomatic formulations that again lead to the same result [64].

Before continuing, we note that modern axiomatic probability theory is often couched in terms of measure theory. This has some benefits, such as unifying considerations of finite and infinite distributions [92]. However, in the work that follows there is no particular need for this formalism, and the mathematical overhead in couching all subsequent work in terms of measures would not add to the reader's understanding. As such, we will consider probability distributions rather than probability measures, whilst being careful to describe the set over which these distributions are defined.

We will also skip a treatment of theories such as fuzzy logic [63], which rather than being based on degrees of plausibility about facts that are either true or false, instead deals with uncertainty in the truth or falsity of the facts themselves.

A.0.1. Cox's Axioms

Let us begin by assuming that the notion of a "degree of belief" exists. This would include both the concepts of true and false but would also encompass values between these two extremes that reflect how much we believe in a proposition given incomplete knowledge. For a proposition X, such as *it is raining*, we denote the belief of X given our current state of information I to be $B(X|I)$. Assume $B(X|I) \in \mathcal{B}$, the set of all beliefs. Denote the value of $B(X|I)$ corresponding to *true* as T and *false* as F. We will specify four properties we would like $B(X|I)$ to have, and from these attempt to ascertain the features of a theory of reasoning under uncertainty.

R1: Degrees of belief are real numbers

- $B(X|I) \in \mathbb{R} \quad \forall X, I.$

As the real numbers are the standard system of measurement for every fundamental unit in physics (length, time, mass etc.), it does seem reasonable to use this set to also measure belief. Jaynes gives this requirement explicitly [52], but we can consider two weaker requirements separately and show they imply $R1$.

R1a: Degrees of belief have a total ordering

- $B(X) > B(Y),\ B(Y) > B(Z) \longrightarrow B(X) > B(Z).$

Intuitively, this is stating that all degrees of belief can be compared. Whilst comparing the chance of rain tomorrow with the likelihood of Mars harbouring life seems incongruent, it is intuitive that the strength of our **beliefs** about them can be compared. R1a immediately rules out any theories which represent belief as a multi-dimensional quantity. Such theories do exist; possibility theory [26] and belief function theories [104] for instance. For an overview of the debate surrounding this requirement, see [64].

R1b: Degrees of belief are infinite

- $|\mathcal{B}| \geq \aleph_0$.

R1b makes sense when we consider the alternative— a finite number of degrees of belief. Imagine there were only N degrees of belief, and we were faced with $N+1$ groups of coins, where the k-th group contained k coins. Intuitively, we would like to assign a different level of belief to the chance that all coins in a group come up heads. But with only N levels of belief we would be forced to assign the same belief to two groups containing different numbers of coins.

R1 satisfies R1a and R1b, but so do countably infinite sets such as the rationals [52]. However, we can make an argument analogous to the N coins to see why a countable number of belief values would make constructing a universal system difficult, especially one that was useful when reasoning about real-valued physical quantities such as length or time.

R2: Beliefs are compatible with propositional calculus

- X is equivalent to X' implies $B(X|I) = B(X'|I)$
- If X is true then $B(X|I) = T$
- $B(X|Y, Z, I) = B(X|(Y \wedge Z), I)$
- If I is consistent and $B(\neg X|I) < T$, then (X, I) is also consistent

Here equivalence means that these two statements have the same truth values (i.e. $X \leftrightarrow X'$ is a tautology). Consistency for a set of information I says that there exist no two propositions X and X' such that $B(X|I) = T$ and $B(\neg X|I) = T$.

R2 states that when two statements are equivalent, our belief in their truth should be equal. If a statement is always true, then it should be given the belief associated with certainty. If two statements are true, we should be able to replace them with their conjunction. And finally, if there is some chance that X is true, then adding it to our known information should not cause our system of reasoning to become inconsistent.

R3: A belief and its negation cannot vary independently.

- There exists a non-increasing function S_0 such that $B(\neg X|I) = S_0(B(X|I))$ for all X and consistent I.

R3 states that once we know on of either $B(X|I)$ or $B(\neg X|I)$, the other is determined (through S_0 or S_0^{-1}). The non-increasing function requirement means that if our belief in X increases, we do not want our belief in $\neg X$ to also increase.

R4: The set of all beliefs is dense.

- If $B(X|I) < B(Y|I)$ then $\exists a \in \mathcal{B}: B(X|I) < a < B(Y|I)$.

There are arguments against R4 [44], but these arguments stem from a desire to define a system over a finite set of possible beliefs (which we addressed in requirement R1b). Such a system would necessarily be tied to the particular finite subset chosen, and hence would be difficult to construe as a general system for reasoning under uncertainty.

A.0.2. Cox's Theorem

On first glance it might appear that $R1 - R4$ are not particularly restrictive, in that there might exist any number of theories that are different from each other but satisfy these axioms. Surprisingly, it is not the case. In fact, **any** system of reasoning for which $R1 - R4$ hold will be isomorphic to probability theory. This is Cox's Theorem.

Therefore, it is without loss of generality that we can describe our degrees of belief in a proposition X given information I as the probability $P(X|I)$, and noting that $(PX|I) = 0$ implies X is certainly false, and $P(X|I) = 1$ is equivalent to the statement X is certainly true.

The implications of Cox's Theorem are significant. First, it states that not using Bayes' theorem for reasoning about uncertainty implies throwing out at least one of R1-R4. It also cements the prior distribution as in integral part of reasoning. Finally, it provides us with a recipe for inference that demands explicit enumeration of our assumptions in the form of prior and likelihood distributions.

A.0.3. Laws of Probability

From the axioms above, we can of course re-create the familiar rules of probability. From here we will explicitly consider random variables rather than individual propositions, noting that the probability distribution $P(X)$ for a random variable X relates to a set of propositions $\{X = x\}, x \in \mathcal{X}$.

Product Rule

For random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, the product rule relates their joint distribution to the conditional by

$$P(X, Y) = P(X|Y)P(Y). \tag{A.1}$$

Sum Rule

The sum rule states that the total probability of all propositions being considered is equal to unity.

$$\sum_n P(X_n|I) = 1. \tag{A.2}$$

Law of total Probability

$$P(X) = \sum_n P(X, Y_n) \tag{A.3}$$

The law of total probability is also referred to as marginalization in the Bayesian literature, because provides a method to 'ignore' unknown variables by placing a prior on them and integrating them out of the expression. This process is made explicit by substituting the sum rule in place of the joint:

$$P(X) = \sum_n P(X|Y_n)P(Y_n) \tag{A.4}$$

A.0.4. Continuous Sets

When the sets $\mathcal{X}$ and $\mathcal{Y}$ are uncountable, there are equivalent expressions for all the above results:

$$P(X, Y) = P(X|Y)P(Y), \tag{A.5}$$

$$\int P(X = x)\,\mathrm{d}x = 1, \tag{A.6}$$

$$P(X) = \int P(X, Y)\,\mathrm{d}Y, \tag{A.7}$$

$$P(X) = \int P(X|Y)P(Y)\,\mathrm{d}Y. \tag{A.8}$$

Appendix B

Full Experimental Results

Table B.1.: Results for the Particle Simulation

Algorithm	2D	3D	4D	5D	6D
PF-500	-0.274 ±0.087	-1.766 ±0.711	-3.277 ±1.683	-3.752 ±1.705	-4.160 ±1.474
PF-1000	-0.072 ±0.088	-1.566 ±0.734	-3.085 ±1.794	-3.546 ±1.672	-3.892 ±1.618
PF-5000	0.389 ±0.098	-1.144 ±0.824	-2.683 ±1.929	-3.037 ±1.984	-3.246 ±1.870
PF-10000	0.585 ±0.106	-0.967 ±0.853	-2.509 ±1.832	-2.859 ±2.041	-2.981 ±1.959
EPF-500	0.188 ±0.001	-1.275 ±0.467	-2.825 ±1.321	-3.329 ±1.272	-3.684 ±1.205
EPF-1000	0.407 ±0.002	-1.082 ±0.504	-2.622 ±1.348	-3.083 ±1.236	-3.376 ±1.167
EPF-5000	0.904 ±0.006	-0.629 ±0.541	-2.192 ±1.396	-2.578 ±1.403	-2.736 ±1.350
EPF-10000	1.112 ±0.009	-0.440 ±0.568	-2.010 ±1.416	-2.366 ±1.526	-2.460 ±1.447
OPF-500	1.230 ±0.128	2.644 ±0.284	1.256 ±0.128	-0.592 ±0.061	-0.748 ±0.076
OPF-1000	1.232 ±0.127	2.613 ±0.280	1.417 ±0.143	-0.625 ±0.063	-0.760 ±0.077
OPF-5000	1.231 ±0.127	2.744 ±0.298	1.677 ±0.169	-0.520 ±0.056	-0.696 ±0.072
OPF-10000	1.232 ±0.128	2.798 ±0.305	1.735 ±0.174	-0.565 ±0.059	-0.702 ±0.072
GPPF-500	-0.256 ±0.463	-1.854 ±1.120	-3.877 ±4.849	-4.822 ±5.160	-4.787 ±4.333
GPPF-1000	-0.081 ±0.522	-1.661 ±1.084	-3.547 ±2.859	-4.582 ±5.697	-4.497 ±3.182
GPPF-5000	0.269 ±0.773	-1.343 ±1.450	-3.150 ±2.676	-4.095 ±4.015	-3.834 ±4.000
GPPF-10000	0.417 ±0.850	-1.224 ±1.661	-2.932 ±2.696	-3.854 ±3.553	-3.609 ±3.374
KBR-N	1.995 ±1.995	-0.172 ±-0.172	-0.336 ±-0.336	0.934 ±0.934	1.256 ±1.256
KBR-NS	1.724 ±1.724	-5.080 ±-5.080	-7.441 ±-7.441	-10.191 ±-10.191	-12.130 ±-12.130
KBR-J	2.510 ±2.510	3.100 ±3.100	4.250 ±4.250	5.324 ±5.324	6.542 ±6.542

Algorithm	7D	8D	9D	10D
PF-500	-6.499 ±3.300	-7.279 ±3.397	-9.665 ±4.778	-10.622 ±4.462
PF-1000	-6.295 ±3.596	-6.956 ±3.171	-9.362 ±4.739	-10.228 ±4.762
PF-5000	-5.792 ±4.192	-6.338 ±3.683	-8.779 ±5.602	-9.552 ±4.751
PF-10000	-5.627 ±4.406	-6.054 ±3.852	-8.502 ±5.993	-9.231 ±5.152
EPF-500	-6.171 ±2.849	-6.779 ±2.661	-9.079 ±4.060	-10.032 ±3.908
EPF-1000	-5.900 ±2.961	-6.458 ±2.900	-8.816 ±4.397	-9.647 ±3.993
EPF-5000	-5.402 ±3.399	-5.867 ±3.150	-8.215 ±4.917	-8.986 ±4.084
EPF-10000	-5.178 ±3.404	-5.545 ±3.399	-8.029 ±4.849	-8.745 ±4.418
OPF500	-1.152 ±0.110	-0.624 ±0.071	-1.155 ±0.113	-1.803 ±0.170
OPF1000	-0.949 ±0.094	-0.527 ±0.065	-0.946 ±0.097	-1.610 ±0.153
OPF5000	-0.820 ±0.083	-0.311 ±0.054	-0.376 ±0.061	-1.167 ±0.116
OPF10000	-0.783 ±0.081	-0.231 ±0.052	-0.244 ±0.057	-1.060 ±0.108
GPPF-500	-7.813 ±8.123	-8.874 ±11.105	-10.874 ±9.402	-12.126 ±13.063
GPPF-1000	-7.475 ±7.994	-8.638 ±10.123	-10.666 ±8.405	-11.782 ±12.107
GPPF-5000	-6.870 ±5.906	-8.127 ±9.954	-9.893 ±6.037	-11.024 ±7.785
GPPF-10000	-6.704 ±6.353	-7.704 ±8.989	-9.574 ±4.849	-10.870 ±8.890
KBR-N	2.318 ±2.318	4.362 ±4.362	4.708 ±4.708	4.627 ±4.627
KBR-NS	-12.970 ±-12.970	-15.220 ±-15.220	-15.902 ±-15.902	-15.861 ±-15.861
KBR-J	7.873 ±7.873	9.273 ±9.273	10.546 ±10.546	11.882 ±11.882

Data Fraction	Random	K-Means	Optimal	Low Rank
0.001	0.265 ± 0.243	−0.014 ± 0.022	−0.022 ± 0.035	−0.894 ± 0.040
0.002	0.081 ± 0.059	−0.025 ± 0.020	−0.077 ± 0.036	−0.818 ± 0.055
0.003	0.087 ± 0.070	−0.270 ± 0.027	−0.328 ± 0.019	−0.850 ± 0.037
0.004	0.084 ± 0.124	−0.391 ± 0.020	−0.397 ± 0.046	−0.806 ± 0.033
0.005	−0.094 ± 0.085	−0.421 ± 0.018	−0.431 ± 0.030	−0.834 ± 0.038
0.006	−0.032 ± 0.107	−0.427 ± 0.016	−0.497 ± 0.013	−0.827 ± 0.079
0.007	−0.119 ± 0.080	−0.429 ± 0.051	−0.517 ± 0.020	−0.829 ± 0.091
0.008	−0.046 ± 0.136	−0.487 ± 0.029	−0.534 ± 0.041	−0.827 ± 0.010
0.009	−0.067 ± 0.069	−0.492 ± 0.032	−0.579 ± 0.046	−0.801 ± 0.080
0.01	−0.096 ± 0.126	−0.590 ± 0.030	−0.636 ± 0.020	−0.815 ± 0.037
0.011	−0.218 ± 0.094	−0.557 ± 0.033	−0.619 ± 0.028	−0.839 ± 0.054
0.012	−0.178 ± 0.108	−0.587 ± 0.019	−0.664 ± 0.045	−0.848 ± 0.032
0.013	−0.147 ± 0.054	−0.610 ± 0.028	−0.658 ± 0.037	−0.855 ± 0.015
0.014	−0.268 ± 0.063	−0.633 ± 0.023	−0.691 ± 0.017	−0.837 ± 0.026
0.015	−0.243 ± 0.172	−0.639 ± 0.032	−0.711 ± 0.056	−0.871 ± 0.034
0.016	−0.189 ± 0.058	−0.599 ± 0.021	−0.707 ± 0.021	−0.854 ± 0.058
0.017	−0.243 ± 0.060	−0.658 ± 0.011	−0.724 ± 0.063	−0.859 ± 0.023
0.018	−0.226 ± 0.098	−0.671 ± 0.030	−0.743 ± 0.034	−0.825 ± 0.019
0.019	−0.234 ± 0.118	−0.681 ± 0.034	−0.761 ± 0.035	−0.831 ± 0.024
0.02	−0.270 ± 0.133	−0.739 ± 0.023	−0.778 ± 0.038	−0.845 ± 0.032

Table B.2.: Toy Sinusoidal data set SNLP results for the various scaling strategies. The exact result was −0.954 ± 0.018.

Table B.3.: Abalone data set Results

Reduced Set Size	SD-MKBR	SMKBR
16	−0.237 ± 0.094	−0.382 ± 0.022
32	−0.259 ± 0.059	−0.430 ± 0.021
64	−0.291 ± 0.021	−0.657 ± 0.306
128	−0.428 ± 0.017	−1.309 ± 0.100
256	−0.684 ± 0.271	−1.670 ± 0.098
512	−1.516 ± 0.334	−2.076 ± 0.176

Table B.4.: Kin40K data set Results

Reduced Set Size	SD-MKBR	SMKBR
16	−0.095 ± 0.049	−0.483 ± 0.088
32	−0.187 ± 0.028	−0.538 ± 0.013
64	−0.227 ± 0.020	−0.516 ± 0.005
128	−0.325 ± 0.022	−0.498 ± 0.009
256	−0.395 ± 0.016	−0.532 ± 0.017
512	−0.481 ± 0.013	−0.558 ± 0.018

Acknowledgements

Always two there are; a master and an apprentice, and my master Fabio Ramos was both wise and powerful. Fabio provided invaluable guidance on all aspects of my doctorate. He helped me determine what directions might be interesting to investigate, what kinds of experiments would effectively illustrate a particular technique, and how to critically examine my own technical writing from the point of view of a reviewer. He was always patient with my programming diversions, and when my work at NICTA limited my time. Most importantly, however, Fabio was unwavering in his support and encouragement, and no matter how overwhelmed I felt, I always came away from a chat with him re-assured. It was an honour and a pleasure to be his student.

Hugh Durrant-Whyte has showed me over the course of my PhD that if I'm tackling a problem that is only semi-impossible, then I have not been sufficiently audacious. I am indebted to him for demonstrating to me that achievement requires as much courage as intelligence and hard work. Hugh showed some of that courage in the degree of trust he placed in me; giving me countless opportunities to present to and interact with an array of great minds from all over the world.

My associate supervisor, Mitch Bryson, provided valuable early guidance and helped ground me in the real world of experiments when I'd been spending too long in Hilbert space. Tim Bailey always provided my PhD cohort and me genuine and honest support, and set the benchmark for clarity of thought, rigour and professionalism to which I continue to aspire.

I am indebted to Kenji Fukumizu of the Institute of Statistical Mathematics for not only producing the algorithm extended in this thesis, but also for welcoming me for a visit to his lab in Tokyo. I gained a lot from the fascinating discussions with him and his students.

Simon O'Callaghan was indispensable as a collaborator in pedestrian velocity field work in Chapter 3. However, both he and the third member of our gang, Alistair Reid, had the much more important job of being living proof that it is possible to finish a PhD. They also helped me understand Gaussian processes, posed hypotheticals, and generally make me eager to get out of bed in the morning and get to the lab.

Similarly, in the earlier stages of my PhD my cohort and fellow cube mates Dan Steinberg, John Vial, Ari Friedman, Ash Bender, Alastair Quadros, Peter Morton and Mark de Deuge always provided a sounding board, many laughs, scintillating lunchtime conversation and invaluable moral support.

I would never have been here were it not for the lessons and inspiration from my teachers and lecturers in college and undergrad; Rae Pottenger, Nick Vonthetoff, Neil Montgomery, and Craig Savage. These teachers fostered my love for maths and science and helped lay the intellectual foundations for the rest of my life. Most of all, Antony Searle first transfixed me with Bayesian probability theory in what was without question the most enjoyable and influential course of my life. Rarely a day goes by in my research when I do not think back to the fundamentals he taught me.

To help address the emotional challenge of the PhD, Nic Boling took me out for breakfast or dinner once a week for 4 years, and never once complained that all I ever did was despair and contemplate whether it was too late to become a baker. Even more impressive, is that this is now the third degree in which he has lent a sympathetic ear and a regular decompression. Similarly my other wonderful friends always made the time seem brighter, and the goal more achievable.

My mother and father have essentially made my education their top priority since I was born, and this document is the fruit of their efforts as well as mine. They, along with my step-mother Kate and my step-father Geoff, were a bottomless well of support, and were unwavering in their belief that I could finish despite my protestations to the contrary. As well as having this same faith in me, my brother Andrew reminded me that life is meant to be enjoyed.

Finally, my partner Jacqualine Myint lived with me through most of the PhD, and so witnessed first hand the crushing weight of anxiety, lack of sleep, and focus on the thesis to the exclusion of all else. I still wonder how she kept her patience, and more than that, provided more love, support and understanding than I thought was possible to give.

Bibliography

[1] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun. Discriminative training of Kalman filters. In **Proceedings of Robotics: Science and Systems**, 2005.

[2] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. **Automation and remote control**, 25:821–837, 1964.

[3] D. Alspach and H. Sorenson. Nonlinear Bayesian estimation using Gaussian sum approximations. **IEEE Transactions on Automatic Control**, 17(4):439–448, Aug. 1972.

[4] M. Alvarez, D. Luengo-Garcia, M. Titsias, and N. Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In **International Conference on Artificial Intelligence and Statistics**, pages 25–32, 2010.

[5] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. **IEEE Transactions on Signal Processing**, 50(2):174–188, 2002.

[6] K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

[7] L. K. Bachrach, T. Hastie, M. Wang, B. Narasimhan, and R. Marcus. Bone mineral acquisition in healthy Asian, Hispanic, black, and Caucasian youth: a longitudinal study. **Journal of Clinical Endocrinology & Metabolism**, 84(12):4702–4712, 1999.

[8] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. **Robotics & Automation Magazine**, (September), 2006.

[9] C. R. Baker. Joint measures and cross-covariance operators. **Transactions of the American Mathematical Society**, 186:273–289, 1973.

[10] H. Barendsen. Introduction to lambda calculus. Technical Report March, 1994.

[11] A. Berlinet and C. Thomas-Agnan. **Reproducing kernel Hilbert spaces in probability and statistics**. Springer, 2004.

[12] C. Bishop. **Pattern Recognition and Machine Learning**. 2006.

[13] E. Bonilla, K. Chai, and C. Williams. Multi-task Gaussian process prediction. In **Advances in Neural Information Processing Systems 20**, pages 153–160, 2008.

[14] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In **Proceedings of the fifth annual workshop on Computational learning theory**, pages 144–152. ACM, 1992.

[15] L. Bottou and O. Bousquet. The Tradeoffs of Large-Scale Learning. **Optimization for Machine Learning**, page 351, 2011.

[16] D. Brunn, F. Sawo, and U. Hanebeck. Nonlinear multidimensional Bayesian estimation with Fourier densities. **Proceedings of the 45th IEEE Conference on Decision and Control**, (1):1303–1308, 2006.

[17] P. Chaudhuri. Nonparametric estimates of regression quantiles and their local Bahadur representation. **The Annals of Statistics**, 19(2):760–777, 1991.

[18] L. Chen and P. Pu. Survey of preference elicitation methods. Technical report, 2004.

[19] V. Chernozhukov, I. Fernández-Val, and A. Galichon. Quantile and probability curves without crossing. **Econometrica**, 78(3):1093–1125, 2010.

[20] T. Cole and P. Green. Smoothing reference centile curves: the LMS method and penalized likelihood. **Statistics in medicine**, 11(10):1305–1319, 1992.

[21] R. Cox. Probability, frequency and reasonable expectation. **American journal of physics**, 14:1, 1946.

[22] M. Deisenroth, C. Rasmussen, and J. Peters. Gaussian process dynamic programming. **Neurocomputing**, 72(7-9):1508–1524, 2009.

[23] A. Doucet, N. D. Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In **Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence**, pages 176–183, 2000.

[24] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. **Handbook of Nonlinear Filtering**, 12:656–704, 2009.

[25] P. Drineas and M. W. Mahoney. On the {N}ystr{ö}m method for approximating a {G}ram matrix for improved kernel-based learning. **The Journal of Machine Learning Research**, 6:2153–2175, 2005.

[26] D. Dubois, H. Prade, H. Farreny, R. Martin-Clouaire, and C. Testemale. **Possibility theory: an approach to computerized processing of uncertainty**. Plenum press New York, 1988.

[27] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. **Robotics & Automation Magazine**, 2006.

[28] J. Freitas, M. Niranjan, A. Gee, and A. Doucet. Sequential Monte Carlo methods to train neural network models. **Neural computation**, 12(4):955–993, 2000.

[29] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. **Journal of the ACM**, pages 1–17, 2004.

[30] K. Fukumizu. Unpublished communications.

[31] K. Fukumizu. Kernel Bayes' Rule: Nonparametric Bayesian inference with kernels. In **NIPS 2012 Workshop on Confluence between Kernel Methods and Graphical Models**, 2012.

[32] K. Fukumizu, F. Bach, and M. Jordan. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. **The Journal of Machine Learning Research**, 5:73–99, 2004.

[33] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf. Kernel measures of conditional dependence. In **Advances in Neural Information Processing Systems**, pages 1–13, 2008.

[34] K. Fukumizu, L. Song, and A. Gretton. Kernel Bayes' rule. In **Advances in Neural Information Processing Systems**, pages 1737–1745, 2011.

[35] K. Fukumizu, L. Song, and A. Gretton. Kernel bayes' rule: Bayesian inference with positive definite kernels. **The Journal of Machine Learning Research**, 14(1):3753–3783, 2013.

[36] Z. Ghahramani. The kin datasets. **URL: http://www. cs. utoronto. ca/delve/data/kin/kin. ps. gz**, 1996.

[37] E. Gilbert and K. Karahalios. Widespread Worry and the Stock Market. In **International AAAI Conference of Weblogs and Social Media**, pages 59–65, 2010.

[38] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. **Mathematical programming**, 27:1–33, 1983.

[39] G. Golub and C. Van Loan. **Matrix computations**, volume 3. JHU Press, 1989.

[40] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. **IEE Proceedings-F Radar and Signal Processings**, 140(2):107–113, 1993.

[41] A. Gretton and K. Fukumizu. A fast, consistent kernel two-sample test. In **Advances in Neural Information Processing Systems**, pages 1–11, 2009.

[42] A. Gretton, K. Fukumizu, and B. Schölkopf. A Kernel Statistical Test of Independence. **MPI Technical Report 168**, 2008.

[43] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding Structure with Randomness : Probabilistic Algorithms for Matrix Decompositions. **SIAM review**, 53(2):217–288, 2011.

[44] J. Halpern. A counter example to theorems of Cox and Fine. **arXiv preprint arXiv:1105.5450**, 10:67–85, 2011.

[45] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. **The Mathematical Intelligencer**, 27(2):83–85, 2005.

[46] X. He. Quantile curves without crossing. **The American Statistician**, 51(2):186–192, 1997.

[47] J. Hensman, U. Sheffield, N. Fusi, and N. Lawrence. Gaussian Processes for Big Data. In **Proceedings of UAI 29**, pages 282–290, 2013.

[48] N. Hjort and S. Petrone. Nonparametric quantile inference using Dirichlet processes. **Advances in statistical modeling and inference: essays in Honor of Kjell A. Doksum**, 1:463, 2007.

[49] N. L. Hjort and S. G. Walker. Quantile pyramids for Bayesian nonparametrics. **The Annals of Statistics**, 37(1):105–131, 2009.

[50] T. Hofmann, B. Schölkopf, and A. Smola. A Tutorial Review of RKHS Methods in Machine Learning. Technical report, 2005.

[51] K. V. Horn. Constructing a logic of plausible inference: a guide to cox's theorem. **International Journal of Approximate Reasoning**, 34(1):3–24, 2003.

[52] E. Jaynes. **Probability Theory: The Logic of Science**. Cambridge University Press, 2003.

[53] A. H. Jazwinski. **Stochastic processes and filtering theory**, volume 63. Academic Press, 1970.

[54] S. G. Johnson. The NLopt nonlinear-optimization package.

[55] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. **Proceedings of the IEEE**, 92(3):401–422, 2004.

[56] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte. A new approach for filtering nonlinear systems. In **Proceedings of the American Control Conference, 1995.**, volume 3, pages 1628–1632, 1995.

[57] P. Kaelo and M. M. Ali. Some Variants of the Controlled Random Search Algorithm for Global Optimization. **Journal of Optimization Theory and Applications**, 130(2):253–264, 2006.

[58] R. E. Kalman. A new approach to linear filtering and prediction problems. **Transactions of the ASME - Journal of Basic Engineering**, 82(Series D):35–45, 1960.

[59] A. Kan and G. Timmer. Stochastic global optimization methods part II: Multi level methods. **Mathematical Programming**, 39:57–78, 1987.

[60] P. Kearey, M. Brooks, and I. Hill. **An introduction to geophysical exploration**. Blackwell series, London, 2009.

[61] A. Khokhlov, E. Mueller, and P. Höflich. Light curves of Type IA supernova models with different explosion mechanisms. **Astronomy and Astrophysics**, 270:223–248, 1993.

[62] N. Kirchner, A. Alempijevic, S. Caraian, R. Fitch, D. Hordern, H. G., G. Paul, D. Richards, S. P. Singh, and S. S. Webb. RobotAssist - a Platform for Human Robot Interaction Research. In **Proceedings of the Australasian Conference on Robotics and Automation 2010**, pages 1–10, 2010.

[63] G. Klir and B. Yuan. **Fuzzy sets and fuzzy logic**. Prentice Hall New Jersey, 1995.

[64] K. Knuth and J. Skilling. Foundations of inference. **Axioms**, 1(3):38–73, June 2012.

[65] J. Ko and D. Fox. GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. **Autonomous Robots**, 27(1):75–90, 2009.

[66] R. Koenker. A note on L-estimates for linear models. **Statistics & probability letters**, 2(6):323–325, 1984.

[67] R. Koenker. **Quantile regression**. Cambridge university press, 2005.

[68] R. Koenker, G. Bassett, and N. Jan. Regression Quantiles. **Econometrica**, 46(1):33–50, 1978.

[69] J. H. Kotecha and P. M. Djuric. Gaussian particle filtering. **IEEE Transactions on Signal Processing**, 51(10):2592–2601, 2003.

[70] J. H. Kotecha and P. M. Djuric. Gaussian sum particle filtering. **IEEE Transactions on Signal Processing**, 51(10):2602–2612, 2003.

[71] S. Kucherenko and Y. Sytsko. Application of deterministic low-discrepancy sequences in global optimization. **Computational Optimization and Applications**, 30(3):297–318, 2005.

[72] R. Landauer. Irreversibility and heat generation in the computing process. **IBM journal of research and development**, (July):183–191, 1961.

[73] J. Langford, R. Oliveira, and B. Zadrozny. Predicting conditional quantiles via reduction to classification. In **Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence**, pages 257–264, 2006.

[74] N. D. Lawrence, M. Seeger, and R. Herbrich. Fast Sparse Gaussian Process Methods: The Informative Vector Machine. In **Advances in Neural Information Processing Systems 15**, 2003.

[75] M. Lázaro-Gredilla. Sparse spectrum Gaussian process regression. **The Journal of Machine Learning Research**, 11:1865–1881, 2010.

[76] C. Lippert, J. Listgarten, Y. Liu, C. M. Kadie, R. I. Davidson, and D. Heckerman. FaST linear mixed models for genome-wide association studies. **Nature Methods**, 8(10):833–835, 2011.

[77] S. Lloyd. Least squares quantization in PCM. **IEEE Transactions on Information Theory**, 28(2):129–137, 1982.

[78] D. J. C. MacKay. **Information theory, inference and learning algorithms**. Cambridge university press, 2003.

[79] L. McCalman and H. Durrant-Whyte. Bayesian filtering with wavefunctions. In **13th Conference on Information Fusion**. IEEE, 2011.

[80] L. McCalman, S. T. O'Callaghan, and F. T. Ramos. Multi-modal Estimation with Kernel Embeddings for Learning Motion Models. In **2013 IEEE International Conference on Robots and Automation**, pages 2845–2852, 2013.

[81] R. V. D. Merwe, A. Doucet, N. D. Freitas, and E. Wan. The unscented particle filter. **Advances in Neural Information Processing Systems**, pages 584–590, 2001.

[82] R. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part II: Convex quadratic programming. **Mathematical Programming**, 44:43–66, 1989.

[83] W. J. Nash. **The Population Biology of Abalone (Haliotis Species) in Tasmania: Blacklip Abalone (H. Rubra) from the North Coast and the Islands of Bass Strait**. Sea Fisheries Division, Marine Research Laboratories-Taroona, Department of Primary Industry and Fisheries, Tasmania, 1994.

[84] Y. Nesterov, A. S. Nemirovskii, and Y. Ye. **Interior-point polynomial algorithms in convex programming**, volume 13. SIAM, 1994.

[85] M. Neumann, K. Kersting, Z. Xu, and D. Schulz. Stacked Gaussian Process Learning. In W. W. H. Kargupta, editor, **Proceedings of the 9th IEEE International Conference on Data Mining**, 2009.

[86] Y. Nishiyama and A. Boularias. Hilbert space embeddings of POMDPs. **arXiv preprint arXiv:1210.4887**, 2012.

[87] S. O'Callaghan, F. T. Ramos, and H. Durrant-Whyte. Contextual occupancy maps using gaussian processes. In **2009 IEEE International Conference on Robotics and Automation**, pages 1054–1060. Ieee, 2009.

[88] S. T. O'Callaghan, S. P. N. Singh, A. Alempijevic, and F. T. Ramos. Learning navigational maps by observing human motion patterns. In **2011 IEEE International Conference on Robotics and Automation**, pages 4333–4340. IEEE, 2011.

[89] C. Paciorek and M. Schervish. Nonstationary covariance functions for Gaussian process regression. In **Advances in Neural Information Processing Systems 16**, pages 273–280, 2004.

[90] V. Paulsen. An introduction to the theory of reproducing kernel Hilbert spaces. Technical Report i, 2009.

[91] T. Poggio and F. Girosi. Networks for approximation and learning. **Proceedings of the IEEE**, 78(9):1481–1497, 1990.

[92] D. Pollard. **A user's guide to measure theoretic probability**, volume 8. Cambridge University Press, 2002.

[93] M. J. D. Powell. **A direct search optimization method that models the objective and constraint functions by linear interpolation**. Springer, 1994.

[94] N. Quadrianto and K. Kersting. Kernel conditional quantile estimation via reduction revisited. **Data Mining, 2009. ...**, 2009.

[95] N. Quadrianto and A. Smola. Kernelized sorting. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 32(10):1809–1821, Oct. 2010.

[96] J. Quiñonero Candela. **Learning with uncertainty-Gaussian processes and relevance vector machines**. PhD thesis, 2004.

[97] J. Quiñonero Candela and C. Rasmussen. A unifying view of sparse approximate Gaussian process regression. **The Journal of Machine Learning Research**, 6:1939–1959, 2005.

[98] C. E. Rasmussen and C. K. I. Williams. **Gaussian Processes for Machine Learning**. The MIT Press, 2006.

[99] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. **Learning for Text Categorization: Papers from the 1998 workshop**, 62, 1998.

[100] A. Schwaighofer and V. Tresp. Transductive and inductive methods for approximate Gaussian process regression. **Advances in Neural Information Processing Systems**, pages 953–960, 2002.

[101] M. Seeger. **Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations**. PhD thesis, 2003.

[102] M. Seeger. Gaussian Processes for Machine Learning. **International Journal of Neural Systems**, 14(2):69–106, Apr. 2004.

[103] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. **Workshop on AI and Statistics**, 9:2003, 2003.

[104] G. Shafer. **A Mathematical Theory of Evidence**. Princeton University Press, 1976.

[105] B. Sholkopf and A. J. Smola. **Learning with kernels**. The MIT Press, Cambridge, MA, 2002.

[106] B. W. Silverman. **Density estimation for statistics and data analysis**. Chapman & Hall/CRC, 1986.

[107] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. **17th International Joint Conference on Artificial Intelligence**, 7:2204–2211, 2006.

[108] A. Smola, A. Gretton, L. Song, B. Schölkopf, and B. Scholkopf. A Hilbert space embedding for distributions. **Algorithmic Learning Theory**, pages 1–20, 2007.

[109] A. J. Smola and B. Schökopf. Sparse Greedy Matrix Approximation for Machine Learning. In **Proceedings of the Seventeenth International Conference on Machine Learning**, pages 911–918. Morgan Kaufmann Publishers Inc., 2000.

[110] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In **Advances in Neural Information Processing Systems 18**, 2006.

[111] L. Song, B. Boots, S. Saddiqi, G. Gordon, and A. Smola. Hilbert space embeddings of hidden markov models. In **Proceedings of the 27th International Conference on Machine Learning**, pages 991–998, 2010.

[112] L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In **Proceedings of the 26th Annual International Conference on Machine Learning**, pages 961–968, 2009.

[113] B. Sriperumbudur, A. Gretton, and K. Fukumizu. Injective Hilbert space embeddings of probability measures. (i), 2008.

[114] I. Steinwart, D. Hush, and C. Scovel. An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels. **IEEE Transactions on Information Theory**, 52(10):4635–4643, Oct. 2006.

[115] K. Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. **SIAM Journal on Optimization**, 12(2):555–573, 2002.

[116] R. Szeliski. **Computer vision: algorithms and applications**. Springer, 2011.

[117] M. Taddy and A. Kottas. A Bayesian nonparametric approach to inference for quantile regression. **Journal of Business & Economic Statistics**, 28(3):357–369, 2010.

[118] I. Takeuchi, Q. Le, T. Sears, and A. Smola. Nonparametric quantile estimation. **The Journal of Machine Learning Research**, 7:1231–1264, 2006.

[119] M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In **International Conference on Artificial Intelligence and Statistics**, volume 5, pages 567–574, 2009.

[120] V. Tresp. Mixtures of Gaussian processes. **Advances in neural information processing systems**, pages 654–660, 2001.

[121] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte. Gaussian process modeling of large-scale terrain. **Journal of Field Robotics**, 26(10):812–840, 2009.

[122] C. R. Vogel. **Computational methods for inverse problems**, volume 10. Siam, 2002.

[123] G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-variance tradeoff and the randomized GACV. In **Proceedings of the 1998 conference on Advances in neural information processing systems II**, pages 620–626, 1999.

[124] C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In **Advances in Neural Information Processing Systems 13**, 2001.

[125] A. G. Wilson and D. A. Knowles. Gaussian Process Regression Networks. In **Proceedings of the 29th International Conference on Machine Learning**, number 1996, pages 599–606, 2012.

[126] M. A. Woodbury. Inverting modified matrices. **Memorandum report**, 42, 1950.

[127] K. Yu and M. C. Jones. Local linear quantile regression. **Journal of the American statistical Association**, 93(441):228–237, 1998.

[128] K. Yu and R. A. Moyeed. Bayesian quantile regression. **Statistics & Probability Letters**, 54(4):437–447, 2001.