Kernel Structure Discovery for Gaussian Process Classification

Nikola Mrkšić
Trinity College

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the Part III of the Computer Science Tripos

May 21, 2014

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Declaration

I Nikola Mrkšić of Trinity College, being a candidate for the Part III in Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 8000

Signed:

Date:

Abstract

Contents

Chapter 1

Introduction

1.1 Building an Automated Statistician

1.2 Contribution

Chapter 2

Background

In this chapter, I will present the main ideas behind Bayesian inference, nonparametric Bayesian models and Gaussian Processes. I will present the main kernel families used for Gaussian Processes and outline the challenges encountered when performing classification using these models.

2.1 Bayesian Inference

In classical probability theory, the probability of an event represents the relative frequency of its occurrence observed after n repetitions of a well-defined random experiment. In such random experiments, an event has only two outcomes: it can either occur or not occur. The sample space ω represents all possible outcomes of the experiment, and an event is described as a subset of this space $\omega_e \subseteq \omega$. If n_e is the number of event occurrences in n trials, the probability of the event is defined as the limit of its relative frequency $\frac{n_e}{n}$ as n tends to infinity. For example, when flipping a coin, the probability of a head outcome corresponds to the number of head outcomes divided by the total number of coin flips performed.

This interpretation, known as the frequentist one, limits us to making observations and drawing conclusions only about those events we can repeat. Events such as 'the world ends tomorrow' make no sense, as there can't be multiple samples of either positive or negative outcomes for this event: it happens only once.

The Bayesian paradigm is more flexible, choosing to regard probabilities as *subjective degrees of belief*. In Bayesian statistics, all types of uncertainty, including our own ignorance, are expressed using the formalism of probability theory [1].

When tackling a problem as a Bayesian, we first choose the likelihood model $\Pr(D \mid \theta, M)$ which we believe can explain the data D in the domain we want to model using some probabilistic model M. We also choose the prior distribution $\Pr(\theta)$ over the unknown parameters θ of the likelihood model chosen. This model and its prior distribution subsume our subjective beliefs about the problem before we have observed any actual data, also known as the evidence. After observing some evidence, we modify our assumptions about the model parameters using Bayes' rule:

$$\Pr(\theta \mid D, M) = \frac{\Pr(D \mid \theta, M) \Pr(\theta \mid M)}{\Pr(D \mid M)}$$

Bayes' rule lies at the core of Bayesian inference. It provides a mathematically rigorous method for adapting our prior assumptions in light of new evidence, allowing us to move from our prior distribution $\Pr(\theta \mid M)$ to the updated posterior distribution $\Pr(\theta \mid D, M)$. If more data becomes available, the posterior can be updated again: the current posterior becomes the prior and is updated using the new evidence D' to arrive at the updated posterior distribution $\Pr(\theta \mid D \cup D', M)$.

The likelihood function $\Pr(D \mid H, M)$ indicates how likely the data D is to be observed assuming the probabilistic model M with parameters θ for our data. Together with the prior $\Pr(\theta \mid M)$, the likelihood model describes the model used. The denominator $\Pr(D \mid M)$ is known as the *marginal likelihood*. It is independent of the parameters θ so it does not affect the posterior distribution $\Pr(\theta \mid D, M)$. This means that the posterior is proportional to the likelihood times the prior.

The intuition behind regarding the parameters as random variables lies in the Bayesian understanding of probability. Even though these values are not random, we model our uncertainty about their 'true' values by assigning some probability mass to all of their potential values. Subsequently, when making predictions for some new instance x, we are in a position to consider all the available evidence by making decisions based on the weighted contributions of all realistic hypotheses θ to our prediction. The weights assigned to different hypotheses come directly from the posterior $\Pr(\theta \mid D, M)$ and are combined with the likelihood model $\Pr(x \mid \theta, M)$:

$$\Pr(x \mid D, M) = \int_{\theta} \Pr(x \mid \theta, M) \Pr(\theta \mid D, M) d\theta$$

This is in direct opposition to the frequentist paradigm, which interprets the data as random samples drawn from some unknown underlying distribution. While a useful assumption to make, it is not actually true. Data is certain, and everything

else in our inference procedure is uncertain: the prior, the likelihood model and the probabilistic model M itself.

The prior on the parameters is the result of our subjective judgement. This is not a weakness, but a strength of the Bayesian paradigm. All models have some assumptions built into them, allowing them to make predictions on the basis of the observed evidence. In the Bayesian paradigm, we are forced to make all of these assumptions explicit. Hence, if our model proves inadequate, it is not the fault of the inference mechanism, but of the flawed assumptions built into it [1].

Another convenient property of Bayesian inference is that the effect of the prior decreases as more data becomes available, with the posterior mostly determined by the new evidence, assuming that the initial prior does not regard the observed evidence as entirely impossible.

The value of marginal likelihood represents the probability of the evidence observed when the model parameters are integrated out, leaving just the dependency of the marginal likelihood on the probabilistic model M used:

$$L \;=\; \Pr(D \mid M) = \int_{\theta} \Pr(D \mid \theta, M) \Pr(\theta \mid M) d\theta$$

Very flexible models can not assign much probability to any data set observed, as they spread their probability mass to a wide array of possible observations. Conversely, overly rigid models are penalised for not being able to assign much probability mass to the evidence across the different parametrisations used. In both cases, this makes the marginal likelihood of these models lower than the marginal likelihood of models well suited to explaining the evidence. Hence, marginal likelihood can be used as the criterion for model ranking: the higher L is, the better the model M is at explaining the evidence D.

The reason for this is that the marginal likelihood criterion subsumes an implicit Occam's razor for achieving a trade-off between the fit quality and model complexity [2, 3]. Since model complexity determines the generalisation ability of the model, Bayesian models are less likely to overfit, that is include the noise in the data as part of the underlying signal, leading to poor performance on new observations.

A fully Bayesian approach would introduce a prior over all possible models M instead of picking one of them. However, choosing the set of all sensible models for a specific dataset is non trivial, and averaging their predictions may be difficult or even intractable. Bayesian nonparametric models are an efficient way to perform this averaging using families of very flexible probabilistic models.

2.2 Bayesian nonparametric models

In statistics, parametric models represent collections of probability distributions which can be described using a finite number of parameters. Given the observed evidence E, one chooses (trains) the model parameters θ, which then contain information extracted from E to be used for subsequent inference. In other words, having trained the parameters, subsequent predictions are independent of the evidence observed [1]:

$$\Pr(x \mid \theta, E) = \Pr(x \mid \theta)$$

For modelling complicated datasets with complex relationships between its features, one has to use increasingly sophisticated models. In general, the larger the number of parameters is, the more flexible the model built becomes.

Nonparametric models reject the notion of using a finite set of parameters to describe the distribution of the data: the larger the data sample is, the more parameters we need to describe it. This idea contradicts the standard Bayesian formulation, where the prior and posterior distributions are defined over a fixed parameter space.

Using a fixed number of parameters limits the complexity of our model and prevents us from capturing all the structure available in large and complicated data sets. Bayesian nonparametric (BNP) models overcome this limitation by transitioning to an infinite-dimensional parameter space, capable of dealing with data sets of arbitrary sizes. In general, the number of parameters utilised grows with the size of the data set available. This means that BNP models are defined over an infinite parameter space but can be evaluated using a finite subset of these to explain any given data sample [4]. Inference is performed by marginalising out the surplus dimensions over the prior, leaving only those dimensions which carry information relevant to explaining the data.

The *nonparametric* attribute in BNP models is somewhat misleading, as not only are these models not without parameters, but they have an infinite number of them. The infinite set of parameters is usually represented using a function or a probability measure [4], where the function . Defining a prior over these requires the use of distributions on functions, measures and other infinite-dimensional objects. These distributions are known as stochastic processes, and many of the resulting BNP models, such as the Dirichlet, Beta and Gaussian processes bear the name of the underlying stochastic process. One can arrive at many of the standard BNP models, including the Gaussian process, by letting the number of parameters tend to infinity in standard parametric models such as the multivariate normal distribution.

Computing the posteriors of the stochastic processes embedded in BNP models is non trivial, due to their infinite number of parameters. However, most BNP models have analytically tractable posteriors. This means that the unused parameters can be integrated out efficiently, leaving us with a finite number of parameters to handle using standard techniques such as Markov chain Monte Carlo, variational inference or message passing algorithms [4]. The techniques chosen to handle inference using these models generally follow the same trade-offs between speed and accuracy as their parametric counterparts.

The theoretical properties of BNP models are harder to express and reason with. Proving convergence and consistency is more difficult, and properties such as the *exchangeability* of the observations must be established. While the theory behind BNP models is more sophisticated and demanding than for the analogous parametric cases, the resulting models are far more flexible and provide an alternative to model selection from a parametrised family of models. Many of the machine learning techniques widely used today are non-parametric: kernel methods, SVMs, Gaussian processes, etc.

2.3 Gaussian Processes

Definition A collection of random variables $\{X_t, t \in T\}$ is a Gaussian process if any finite dimensional subset of these random variables has a consistent joint Gaussian distribution [5].

The simplest way of thinking about the Gaussian process is as of an infinite-dimensional generalisation of the multivariate Gaussian distribution. The Gaussian distribution is fully specified by its mean vector μ and covariance matrix Σ. For the sake of building intuition, one may regard any function $\mathbf{f}$ as an infinitely long vector, indexed by the elements of the index set T. Then, for any choice of distinct values $\{t_1 \dots t_k\} \subset T$, the *random vector* $\mathbf{f} = (f_{t_1}, \dots, f_{t_k})^\top$ has a multivariate distribution with mean $\mathbb{E}(\mathbf{f}) = \mu$ and covariance matrix $\mathrm{cov}(\mathbf{f}, \mathbf{f}) = \Sigma$, that is:

$$\mathbf{f} \;=\; (f_{t_1}, \dots, f_{t_k})^\top \;\sim\; \mathcal{N}(\mu, \Sigma)$$

Letting the length of this vector tend to infinity, we arrive at the Gaussian process: the finite random vector $\mathbf{f}$ becomes a function and the mean vector and the covariance matrix become functions $m(x)$ and $K(x_1, x_2)$. We will denote draws from the resulting stochastic process as:

$$\mathbf{f} \sim GP(m, K)$$

where the values $f(t_1), \ldots, f(t_k)$ are now joint Gaussian random variables for any finite subset of T. A common assumption is that these random variables have mean zero, which simplifies calculations without loss of generality. This assumption allows the properties of the Gaussian process to be fully specified by the covariance function K. In practice, this can be implemented by centring the examples in the data set during the pre-processing stages.

Given $GP(m, K)$ and a finite set of N observations $\mathbf{f} = \{f_1, \ldots, f_N\}$, the mean vector and the covariance (Gram) matrix of the joint Gaussian distribution of these observations are given by:

$$\mu_i = m(f_i) \ , \ \Sigma_{i,j} = K(f_i, f_j)$$

2.3.1 Kernels for Gaussian Processes

Gaussian processes specify distributions over functions. This means they can be used as Bayesian priors to express our beliefs about the functions we modelling. A GP is fully specified by its covariance function (*kernel*). The choice of the kernel thus determines the properties of functions described by our prior [3]. Covariance functions characterise correlations between different random variables in the stochastic process. To act as a kernel, a function K must be *positive definite*, meaning that for any finite index set $\{x_1, \ldots, x_n\}$, the resulting $n \times n$ Gram matrix $\mathbf{\Sigma}$ satisfies $v^\top \mathbf{\Sigma} v \geq 0$ for all non-zero vectors v of length n.

One such kernel is the *squared exponential* (SE) covariance function:

$$K(\mathbf{x}, \mathbf{x}') \ = \ \sigma_0^2 \exp\left(\ - \frac{(\mathbf{x} - \mathbf{x}')^2}{2\lambda^2} \right)$$

This kernel makes the (random) values of points close in the input space correlate strongly, whereas distant points are not correlated at all. It serves to represent a family of very smooth, infinitely differentiable functions. The *signal variance* σ_0 and the *characteristic lengthscale* λ are hyperparameters of the covariance function. Figure 2.1 shows two functions drawn from GP priors with SE kernels. The lengthscales determine the length of the trends kernels are able to capture. Longer values allow kernels to represent long-distance dependencies, but reduce their ability to capture short-term trends in the data.

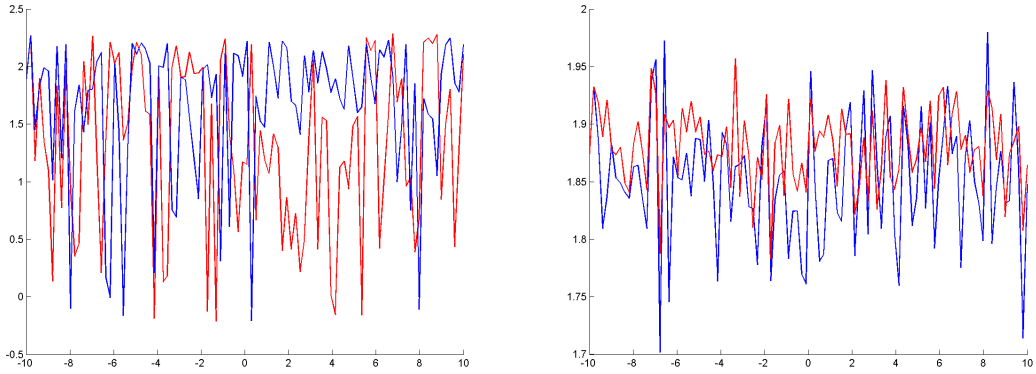


Figure 2.1: Functions drawn from GP priors with SE covariance functions. Lengthscales of 1.0 (left) and 20.0 (right) reflect the length of the trends the kernel is able to capture.

Other classes of kernels frequently used are the Matérn, Linear, Periodic, Rational Quadratic, change-point and many others [3, 6]. We will focus mainly on the SE kernels, which are used as the building blocks of our structure discovery algorithm.

Hyperparameters are 'parameters' of the GP prior itself. They can be learnt from the data by maximising the value of the marginal likelihood. The *hierarchical* nature of the kernel allows one to specify vague information about the prior [5]. For example, we may believe that an SE kernel is appropriate for a given dataset, but have no idea about the right lengthscale or signal variance to use. Thus, these unknown parameters specify a family of models to chose one from. As explained in Section 2.1, the marginal likelihood of these models can be used to discriminate between them, relying on the implicit Bayesian Occam's razor [2]. The dataset-specific hyperparameters maximising this value can be inferred from the data using techniques such as conjugate gradients or gradient descent. Details of this procedure will be examined in subsequent sections.

2.4 Gaussian Process Regression

In the case of regression, we are modelling some function $\mathbf{f}$ given a set of n noisy observations $(X, \boldsymbol{y})$. Assume additive white noise, such that $\mathbf{y} = \mathbf{f} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$. The likelihood, also known as the *noise model*, becomes:

$$\Pr(\boldsymbol{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$$

This likelihood function, together with the GP prior, makes the task of predicting function values $\mathbf{f_T}$ for a set of test locations X_t computationally feasible. The

observations $(X, \boldsymbol{y})$ and the values of the function at test data locations have a joint Gaussian distribution [3]:

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) \triangleq \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K + \sigma_n^2 I & K_* \\ K_*^\top & K_T \end{bmatrix}\right)$$

For notational convenience, replace the covariances $K(X, X)$ and $K(X_*, X_*)$ with K and K_T, respectively. We also abbreviate the cross-covariance $K(X, X_*)$ to K_*. By applying the standard formula for conditioning a joint Gaussian distribution [5], we obtain the expression for the predictive distribution [3]:

$$\mathbf{f}_* \mid X, \mathbf{y}, X_* \sim \mathcal{N}\left(K_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y} \;,\; K_T - K_*^\top (K + \sigma_n^2 I)^{-1} K_* \right)$$

The equation above shows the noise-free predictive mean and variance of the GP at the test points X_*. To obtain the predictive distribution for $\mathbf{y}_*$, we just need to add σ_n^2 to the predictive variance in the equation above. $K_T = K(X_*, X_*)$ is the prior covariance, from which we subtract the strictly positive term $K_*^\top (K + \sigma_n^2 I)^{-1} K_*$ to obtain the covariance of the posterior distribution. This reduction in variance reflects the fact that conditioning on the data gave us additional information which reduced our uncertainty about the underlying function [3].

The complexity of the algorithm computing these values is $O(n^3)$: it is dominated by the inversion of $K + \sigma_n^2 I$. The Cholesky decomposition is typically used instead of inverting the matrix directly, as it has lower overhead and provides better numerical stability. Having computed the inverse, the costs of predicting the mean and the variance for a single test point x_* are $O(n)$ and $O(n^2)$, respectively.

2.4.1 Learning the Hyperparameters

The choice of adequate kernel hyperparameters is essential for adapting the covariance function to the data set provided. Popular machine learning models such as support vector machines or splines use cross-validation to learn hyperparameters. An advantage of Gaussian processes is that the higher values of the marginal likelihood reflect better models [2, 3, 5]. Therefore, the 'best' hyperparameters are those which maximise the marginal likelihood (also known as *evidence*).

In the regression setting, we can use the fact that $\boldsymbol{f} \mid X \sim \mathcal{N}(\mathbf{0}, K)$ and that $\boldsymbol{y} \mid \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}, \sigma_n^2 I)$ to conclude that $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$. Using this result, the (log) marginal likelihood can be expressed as:

$$L \;=\; \log \Pr(\boldsymbol{y} \mid X) \;=\; -\frac{1}{2}\boldsymbol{y}^\top (K + \sigma_n^2 I)^{-1}\boldsymbol{y} - \frac{1}{2}\log\left|K + \sigma_n^2 I\right| - \frac{n}{2}\log 2\pi$$

The first term in the equation above is a negative quadratic measuring how well the kernel fits the target values $\boldsymbol{y}$. The second term penalises overly complex models, ensuring that the marginal likelihood includes an automatic Occam's razor effect for achieving a trade-off between the quality of fit and predictive power on new data. This is important because the function learned could get arbitrarily close to the evidence by setting the signal variance σ_n to 0. That function would model the noise as part of the underlying signal, leading to poor predictive performance. The third term is a normalising constant which does not affect model selection.

The automatic Occam's razor effect greatly simplifies the training process, as there are no parameters balancing model fit and model complexity to be set manually. Training a GP reduces to finding the values of model hyperparameters which maximise the marginal likelihood given the observed data. Since we can find partial derivatives of the marginal likelihood with respect to the hyperparameters, we can use standard gradient based optimisation procedures such as conjugate descent [5]. As described in subsequent Chapters, this is usually a non-convex optimisation problem: different optima correspond to different interpretations of the data. This means that the optimisation procedure should be re-run with different starting hyperparameters to avoid particularly bad local optima.

2.5 Gaussian Process Classification

In binary classification, examples have two different labels: $y_i \in \{-1, 1\}$. There is no underlying real-valued function $\mathbf{f}$ to model. In order to tackle this problem using Gaussian processes, we will assume the existence of a *latent function* $\mathbf{f}$ which governs the class membership of the data. To obtain the prior class distribution at test point $\mathbf{x}_*$, we convert the value of $\mathbf{f}$ at that point into a valid 'class 1' probability using the *response function* σ:

$$\pi(\mathbf{x}_*) \triangleq \Pr(y_* = +1 \mid \mathbf{x}_*) = \sigma(\mathbf{f}(\mathbf{x}_*))$$

As the sigmoid function σ, we could use the logistic sigmoid $\sigma(x) = 1/(1 + e^{-x})$ or the cumulative Gaussian distribution $\Phi(x)$. These functions 'squash' their arguments, which can be arbitrary real numbers, to values between 0 and 1. As the

output of $\mathbf{f}(\mathbf{x})$ can be an arbitrary real number, this step is required to give a valid probabilistic interpretation to the values given by the prior class distribution π.

$\pi(\mathbf{x})$ represents our subjective belief that the example at $\mathbf{x}$ belongs to class 1. For binary classification, the probability of the other class is simply $1 - \pi(\mathbf{x})$. If we let $f_i = \mathbf{f}(x_i)$, we can concisely represent the likelihood (for a single example) as:

$$\Pr(y_i \mid x_i, \mathbf{f}) = \Pr(y_i \mid \mathbf{f}(x_i)) = \sigma(y_i \mathbf{f}(x_i)) \triangleq \sigma(y_i f_i)$$

The multi-class version is harder, as the *softmax function* replaces the sigmoid [3].

We are not particularly interested in the actual values of $\mathbf{f}(x_*)$: we care only about the class membership of points at x_*. The latent function is there to act as the Gaussian prior to be integrated out when performing inference. The difficulty lies in the fact that the likelihood model is no longer Gaussian. If there are n observed examples $\{X, \mathbf{y}\} = \{(\mathbf{x_i}, y_i), i \in 1, \ldots, n\}$, and we adopt the notation that $y_i = i$ for $i = \pm 1$, we can express the joint likelihood $\Pr(\mathbf{y} \mid \mathbf{f})$ as:

$$\Pr(\mathbf{y} \mid \mathbf{f}) \quad = \quad \prod_{i=1}^{n} \Pr(y_i \mid f_i) \quad = \quad \prod_{i=1}^{n} \sigma(y_i f_i)$$

If we assume a zero-mean $GP(0, K)$ over the latent function $\mathbf{f}$ and let $\mathbf{\Sigma} = K(X, X)$, the posterior distribution of the latent function can be expressed as:

$$\Pr(\mathbf{f} \mid X, \mathbf{y}) = \frac{\Pr(\mathbf{f} \mid X) \Pr(\mathbf{y} \mid \mathbf{f})}{\Pr(\mathbf{y} \mid X)} = \frac{\mathcal{N}(\mathbf{f} \mid 0, \mathbf{\Sigma})}{\Pr(\mathbf{y} \mid X)} \prod_{i=1}^{n} \sigma(y_i f_i)$$

This expression contains the marginal likelihood $\Pr(\mathbf{y} \mid X)$, which becomes:

$$\Pr(\mathbf{y} \mid X) = \int \Pr(\mathbf{y} \mid \mathbf{f}) \;\; \Pr(\mathbf{f} \mid X) \; \mathrm{d}\mathbf{f} = \int \left(\prod_{i=1}^{n} \sigma(y_i f_i) \right) \mathcal{N}(\mathbf{f} \mid 0, \mathbf{\Sigma}) \; \mathrm{d}\mathbf{f}$$

The posterior and the marginal likelihood involve a product of a Gaussian with a product of sigmoid functions, which makes both of these expressions analytically intractable. The predictive distribution $\Pr(\mathbf{y}_* = 1 \mid X, \mathbf{y}, x_*)$ and the gradients of the marginal likelihood required to fit the model hyperparameters during the training process become intractable as well [3, 7, 8].

This means one has to resort to approximations in order to perform classification using Gaussian Processes. Most of the known approaches revolve either around Markov chain Monte Carlo sampling or Gaussian approximations to the posterior distribution, such as Laplace's method or Expectation Propagation [8].

2.5.1 Approximative Inference for GP Classification

Markov chain Monte Carlo sampling methods can be used to obtain exact estimates of the required distributions. However, these estimates require very long running times to converge [7]. As such, they are less applicable in practice, but they provide a gold standard to compare the analytic approximations against.

The analytic approximations such as Laplace's method or Expectation Propagation compute Gaussian approximations to the posterior $\mathbf{q}(\mathbf{f}_* \mid X, \mathbf{y}) = \mathcal{N}(\mathbf{f} \mid \mathbf{m}, \mathbf{A})$. The distribution of the latent function $\mathbf{f}_*$ at test point x_* becomes:

$$\Pr(\mathbf{f}_* \mid X, y, x_*) \approx \mathbf{q}(\mathbf{f}_* \mid \mathbf{m}, \mathbf{A}) = \mathcal{N}(\mu_*, \sigma_*^2)$$

$$\mu* = \mathbf{k}_*^\top \mathbf{K^{-1}} \mathbf{m} \quad \text{and} \quad \sigma_*^{\mathbf{2}} = \mathbf{k}(\mathbf{x}_*, \mathbf{k}_*) - \mathbf{k}_*^\top (\mathbf{K^{-1}} - \mathbf{K^{-1}} \mathbf{A} \mathbf{K^{-1}}) \mathbf{k}_*$$

$\mathbf{k}_*$ denotes the vector of prior covariances between the training data X and x_*. $\mathbf{m}$ and $\mathbf{A}$ are parameters of the approximate posterior $\mathbf{q}$ to be learned using Laplace's method or Expectation Propagation [8]. If we use the cumulative Gaussian distribution Φ as the sigmoid function, we can obtain an analytic expression for the predictive distribution as well:

$$\mathbf{q}(y_* = 1 \mid X, \mathbf{y}, \mathbf{x}_*) \;=\; \int \Phi(\mathbf{f}_*)\, \mathcal{N}(\mathbf{f}_* \mid \mu_*, \sigma_*^2) \mathrm{d}\mathbf{f}_* \;=\; \Phi\left(\frac{\mu_*}{\sqrt{1 + \sigma_*^2}} \right)$$

2.5.2 The Laplace Approximation

Laplace's method finds the approximate value of the unnormalised log posterior by performing a second-order Taylor expansion around the mode of this distribution. The approximate log posterior at the test points can be expressed as:

$$\log \mathbf{q}(\mathbf{f} \mid X, \mathbf{y}) \approx \log \mathbf{q}(\mathbf{m}) - \frac{1}{2} (\mathbf{m} - \mathbf{f})^\top \mathbf{A^{-1}} (\mathbf{m} - \mathbf{f})$$

$$\text{where} \quad \mathbf{m} = \underset{f \in \mathbf{R^n}}{\operatorname{argmax}} \log \mathbf{q}(\mathbf{f} \mid X, \mathbf{y}).$$

The likelihood and the prior are both log-concave and unimodal functions, which means $\mathbf{m}$ can be approximated using the iterative Newton-Raphson method:

$$\mathbf{f} \;\leftarrow\; \mathbf{f} \;-\; (\; \nabla\nabla_{\mathbf{f}} \log \mathbf{q}(\mathbf{f}) \;)^{-1} \; \nabla_{\mathbf{f}} \log \mathbf{q}(\mathbf{f})))$$

This procedure usually converges to the mode within a few iterations. The covariance matrix of the approximate posterior distribution can be expressed as $\mathbf{A} = \nabla\nabla_{\mathbf{f}}(\log \mathbf{q}(\mathbf{m}))^{-\mathbf{1}}$ [8]. The closed-form expression for $\mathbf{q}(\mathbf{f} \mid X, \mathbf{y})$ can be used to obtain a closed-form approximation to the log marginal likelihood $\log \Pr(\mathbf{y} \mid X)$:

$$\log \Pr(\mathbf{y} \mid X) \approx \log \mathbf{q}(\mathbf{y} \mid X) = \int \Pr(\mathbf{y} \mid \mathbf{f})\ \mathbf{q}(\mathbf{f} \mid X)\ \mathrm{d}\mathbf{f} =$$

$$= \log \mathbf{q}(\mathbf{m}) + \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\mathbf{A}|$$

The derivatives of the marginal likelihood with respect to the GP hyperparameters (omitted from all the expressions above) have closed-form solutions, so they can be used for hyperparameter optimisation [7].

2.5.3 Expectation Propagation

Expectation Propagation goes further than Laplace's method: the approximate posterior distribution constructed tries to match the first two moments of the actual distribution. The fundamental problem with Laplace's approximation is that the posterior mode becomes less representative of the posterior mean as the number of dimensions increases [7, 8].

The probability mass is concentrated in a thin shell (far) away from the mode: the posterior mean moves towards this high density region, but the posterior mode stays closer to the origin. This means that Laplace's method produces overly cautious predictive distributions which are often inaccurate even near the training locations. As shown in previous work by Kuss and Rasmussen, EP tends to outperform Laplace's method on real-world data sets [7, 8].

The two methods have the same asymptotic complexity, dominated by several iterations of matrix inversions which take $O(n^3)$ time. However, the computational overhead for Expectation Propagation is substantially higher, making Laplace's method much faster in practice. The difference in speed and performance between these two methods will be analysed in subsequent Chapters.

Chapter 3

Related Work

In this chapter, I will present recent research on kernel learning and outline the ideas behind Additive Gaussian Processes and the work on kernel structure discovery for GP regression, which was the main inspiration for this Research Project.

3.1 Kernel learning

Nonparametric models can learn arbitrarily smooth functions from the training data. As the number of dimensions D increases, the basic versions of these algorithms need to use very large data sets to extract the underlying structure. Even if such data sets are available, their size makes inference using nonparametric methods such as GPs computationally infeasible. This means that *additional constraints* on the structure of the models need to be imposed to make inference with high dimensional data sets possible.

Generalised additive models (GAM) assume that the response function y is a transformed sum of functions defined on individual dimensions. If g is the *link function* and $\mathrm{f_i}$ potentially different functions operating on different input dimensions, then:

$$g(y) \;=\; \mathrm{f_1}(x_1) + \ldots + \mathrm{f_D}(x_D)$$

Even though this is a very strict constraint on the model structures permitted, the models constructed often perform well and are able to provide interpretability by directly showing the response function's dependence on each of the dimensions.

The problem of constructing kernels using weighted sums of base kernels has been studied extensively [9]. Most of these algorithms find the optimal solution in poly-

nomial time by specifying the component kernels and hyperparameters beforehand. Kernel methods, on the other hand, allow the response function to depend on all the input dimensions simultaneously:

$$y \;=\; \mathrm{f}(x_1, \ldots, x_d)$$

Examples of such methods include Gaussian Processes with Squared Exponential kernels. More specifically, the SE Automatic Relevance Determination (ARD) kernel allows the input dimensions to vary across different dimensions, as opposed to the previously defined isotropic kernel. The SE ARD kernel function is:

$$K(\mathbf{x}, \mathbf{x}') \;=\; \sigma_0^2 \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_i - x_i')^2}{\lambda_i^2}\right)$$

Other popular approaches learn kernel embeddings into low-dimensional spaces and use fixed kernel structures in those spaces [10]. Deep neural networks have also been applied to learning embeddings [11]. Even though deep networks are not formally nonparametric models, they can express very flexible functions, as the number of units in their layers often exceeds the number of (hyper)parameters in traditional BNP models. The success of these approaches relies on inferring structure from the density $\Pr(x)$, so is most naturally applied to unsupervised learning problems.

The work presented in this Dissertation builds on and extends previous research on automatic kernel construction for the purposes of classification. In this context, most of the interesting structure to be captured lies in $\mathrm{f}(\mathbf{x})$, making the methods geared towards unsupervised learning less relevant.

3.2 Additive Gaussian Processes

Duvenaud et al. (2011) generalise GAM and the GP SE kernel models by constructing a GP kernel which implicitly sums over *all possible products* of one-dimensional base SE kernels [12]. The *additive kernels* which sum over all one-dimensional base kernels and over *all products* of n distinct base kernels are given by:

$$k_1(\mathbf{x}, \mathbf{x}') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x_i') \quad \text{and} \quad k_n(\mathbf{x}, \mathbf{x}') = \sigma_n^2 \sum_{1 \le i_1 < \ldots i_n \le D} \left[\prod_{d=1}^{n} k_{i,d}(x_{i_d}, x_{i_d}')\right]$$

The nth additive kernel k_n represents the sum of all possible n-way interactions

of the input dimensions. σ_n represents the variance hyperparameter of each order of interaction. Unlike other approaches to kernel learning which require manual specification of the kernel structure, Additive GPs only have to fix the n functions $\mathrm{f}_1 \ldots \mathrm{f}_\mathrm{n}$ to be used as the basis functions for each of the dimensions.

Each k_n term contains $\binom{D}{n}$ different product terms. The full additive GP is defined as the sum of kernels for all possible orders of interaction:

$$k_{\text{full}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{D} k_i(\mathbf{x}, \mathbf{x}')$$

The full additive kernel includes any potential interaction between any number of basis functions. The model is very general: the k_1 summation represents the GAM model and the k_D summation represents the GP SE ARD kernel. Its hyperparameters consist of the additive kernel variances $\sigma_1 \ldots \sigma_D$ and the hyperparameters of each of the n basis functions (lengthscales, signal variances, etc). These can be learned by optimising the marginal likelihood of the training data [12].

The $\sigma_1 \ldots \sigma_n$ hyperparameters are very informative: they show what proportion of the target function's variance comes from each order of interaction. Low order interactions between different dimensions can be represented visually. First-order terms represent individual basis functions which we can plot after learning the hyperparameters. For two-way interactions, we can plot the posterior means of the individual second-order terms. For higher level terms, plotting the functions learned is not straightforward, and interpreting the higher level interactions between features becomes increasingly hard.

If we consider all interactions of order up to n, the Additive GP contains $O(2^n)$ terms. For $n \coloneqq D$ the summation would be intractable if approached naively. It is possible to reduce the time complexity to $O(D^2)$ using the Newton-Girard formulae for computing *elementary symmetric polynomials*. This makes the procedure for training and using additive GPs tractable. Low-level interactions prove to be the most relevant for modelling and providing interpretability. This means one can trade small loses in predictive accuracy for substantial improvements in speed while retaining the interpretable components of the additive kernel [12].

Additive GPs are applicable to both classification and regression problems. Standard SE and GAM kernels fail to capture structure which involves patterns longer than the lengthscales of the kernels they use. Additive GPs are able to capture non-local structure using the higher-order interactions and use this information for extrapolation to distant parts of the input space [12].

Additive GPs have been compared to GAM models, GP SE models, Linear and Logistic regression, as well as to other state of the art methods such as Hierarchical Kernel Learning (HKL). In the cases when the data was explained mostly by low-order interactions, Additive GPs outperformed the other methods. When the higher levels of interactions were more dominant, the performance of Additive GPs resembled that of the GP SE model, which makes sense since these kernels represent a subset of the kernels included in the full Additive GP model.

3.3 Kernel structure discovery for GP regression

In spite of the growing popularity of nonparametric Bayesian methods, the choice of kernels is still left to the individual user. This is a major obstacle to making these models easily accessible to non-practitioners without a machine learning background. Additive GPs remove the need for the users to specify the exact kernel structure or its hyperparameters. This model requires manual specification of the basis kernels to make the full kernel construction automatic. Setting these functions requires expert knowledge, so the next step in automatizing kernel construction is to eliminate the need for users to choose the basis kernels themselves.

Duvenaud, Lloyd et al. (2013) tackle this problem by defining a *grammar* over the kernel structures [6]. The base kernels of this grammar are one-dimensional kernels defined on each of the input dimensions, such as SE_1, RQ_2, $\textsc{Per}_3$, where the subscript indicates the dimension the given base kernel applies to. In the original work, four kernel families are considered: SE, RQ, Lin and Per. Subsequent work includes the changepoint kernel as well [13]. Any algebraic expression which combines these kernels using the $+$ and $\times$ operators defines a valid kernel structure.

This grammar includes all models which can be expressed using Additive GPs, GAM and SE models. It can also express other models such as $\mathrm{SE}_1 \times (\mathrm{SE}_3 + \mathrm{SE}_4)$. This kernel is not equivalent to $\mathrm{SE}_1 \times \mathrm{SE}_3 + \mathrm{SE}_1 \times \mathrm{SE}_4$: the first model has three, and the second model has four lengthscale hyperparameters.

Next, we need to define the procedure which explores this grammar to find the best composite kernel for the given data set. The search procedure starts by evaluating all one-dimensional base kernels and assigning them a *score value*.

Marginal likelihood is used as the model selection criterion. Since we are comparing kernel families defined by composite kernels, we need to integrate out the model hyperparameters. This integral is intractable and is approximated by the Bayesian

Information Criterion, which penalises models that overfit by introducing too many hyperparameters, in this case corresponding to large numbers of kernels [14].

Once the kernel structure (corresponding to some expression $\mathcal{A}$) with the best BIC value is identified, the search expands this structure by trying to add or multiply each of its subexpressions $\mathcal{S}$ with all base kernels $\mathcal{B}'$. The procedure also tries to replace any base kernel $\mathcal{B}$ in $\mathcal{A}$ with any other base kernel $\mathcal{B}'$. Of all the kernel structures constructed, the one with the best BIC value is chosen as the next kernel to expand. This process is repeated until the search can no longer improve on the best BIC value by using a more sophisticated kernel structure.

The Additive GP model implicitly sums over an enormous number of kernels. In most cases, few kernels (often low-order interactions) account for most of the contribution to the latent function. This means that a much 'lighter' model can often be used in place of the Additive GP model. Indeed, GP Structure Search is able to outperform the GP Additive model using much smaller kernels which at the same decompose into easily interpretable components [6, 13].

This method provides a data-driven method to choose the kernel structure for any specific problem. The space of kernels that the search can construct is extremely rich, and there is no manual parameter tuning on the side of the user. In spite of the very general kernel grammar, the use of the BIC approximation for model selection raises important questions about the validity of this procedure. Devising a well-founded guiding criterion would be a major step in dispelling the existing skepticism about the potential of this automatic kernel discovery procedure.

GP Structure Search for regression achieved formidable results, and the subsequent work on automatic report generation has attracted a lot of interest in the research community. The main goal of this Research Project was to investigate whether automatic kernel construction is possible for classification problems as well.

Chapter 4

Kernel Structure Discovery for GP Classification

This chapter presents the work done on extending the kernel structure search procedure to the classification case. It first discusses the algorithm and the design choices involved in developing for performing kernel structure discovery in these problems. It provides a proof of concept for the algorithm by demonstrating that it is able to uncover the underlying true kernel for synthetic data sets. The last part of this Chapter deals with applying the search procedure to real-world data sets and discusses the difficulties overcome in achieving state of the art performance.

The Difficulties with Classification

In the classification setting, the likelihood functions become non-Gaussian and the binary data labels decrease the information content of the data sets. Finding interpretable components and patterns in the data becomes much harder than in the case of the time series data sets considered in the existing work on regression. Finally, the use of the Bayesian Information Criterion as a model selection criterion becomes even more problematic, as the number of effective hyperparameters becomes harder to estimate in the classification setting.

4.1 Building the Models

This section presents the kernel grammar chosen and the search operators used to define it. Three different methods for performing approximative inference are

analysed and the implementation details of the hyperparameter estimation are discussed.

4.1.1 Defining the Kernel Grammar

Adding or multiplying with a base kernel. Discuss what adding two kernels or multiplying them accomplishes. Show SE1, SE2, individual plots, then SE1 + SE2, SE1 X SE2 plots.

Compare with the regression case, explain why the RQ and Periodic kernel make less sense, and justify why a pure SE grammar was used (together with a constant mean function to deal with class imbalance in the data sets).

Discuss sum of products vs grammar which can express everything. The grammar can represent exactly the terms in an Additive GP with all base functions set to SEs. Sum of products achieves lower accuracy, but makes the search much faster and the composite kernels much more interpretable.

Should I include (and how much) implementation details: languages used, parallelisation on fear, details of GPML and the infrastructure of the system built...

4.1.2 Approximative Inference

EP vs Laplace vs VB in GPML. EP much slower than Laplace, difference in performance non-existant in all experiments: pure SE grammar makes hyperparameter fitting much easier than was the case in previous work on regression. VB bug?

Reiterate that marginal likelihood is used as the criterion for model selection. Explain that the hyperparameters are found using conjugate gradients in GPML-quickly explain this procedure.

Discuss the computational complexity: suited to use of small training sets, as the training complexity is $O(n^3)$. With larger data sets, we can still fit the hyperparameters faster using subsampling, and this will be heavily exploited in subsequent work, but the final fitting steps take $O(n^3)$ and as such prevent usage of very large data sets available today.

Subsampling the training data to make the hyperparameter fitting faster: perform 300 iterations (a good rough guideline is the number of hyperparameters * 50) with 250 points randomly sampled from the data set, then perform an additional 50 iterations with the full data set.

Due to the smoothness of the SE kernels, this algorithm converges to the same hyperparameters as the minimization that uses the entire data set for the whole process.

Discuss parallelisation and the use of the cluster. Discuss and add a table of running times and numbers of restarts performed.

4.2 Searching for the Model

Discuss the greedy search and backtracking, as well as feature reduction (all implemented).

4.2.1 Bayesian and Akaike Information Criteria

For high values of the number of data points n, the Bayesian Information Criterion can be approximated using the following expression:

$$BIC = -2 \ln L + k \ln(n)$$

where L is the log marginal likelihood of model M on the data set $\mathbf{X}$: $\Pr(\mathbf{X} \mid \theta, M)$, where the hyperparameters θ are set to achieve the maximal value of L using conjugate descent. k is the number of *effective hyperparameters* of the model: estimating this number is non-trivial, and a significant portion of our work consisted of examining different approaches to approximating this figure.

The Akaike Information Criterion (AIC) penalises the number of parameters less than BIC does, no longer taking into account the actual size of the data set:

$$AIC = -2 \; lnL + 2k$$

4.2.2 The Number of Effective Hyperparameters

A major problem for applying BIC to classification is that estimating the number of effective hyperparameters is hard: not all of them affect the classification boundary equally. The only thing important for classification is the zero-crossings of the function (where it is greater, and where it is smaller than 0, for the purposes of prediction). Hence, smaller lengthscales are the most important factor, as they

determine where the predictive mean's zero-crossings exactly are. Very large length-scales equate to multiplication with constant functions, so they have less impact on the classification boundary. Signal variances are useful for computing the predictive variances and determining the relative contribution of each term, but only the predictive means matter for determining the target label. Not counting signal frequencies adds a quick boost to the quality of BIC as a model discriminator: we will refer to this 'modified' criterion as *BIC light*.

4.2.3 Cross-validated Training Accuracy

Discuss the use of cross-validated train accuracy - how it may be expected to provide us with the ceiling performance, but in the end it doesn't improve much on BIC light and likMix models.

Include algorithmic package pseudo code to explain the process.

Show traces of the search, showing how it can build more and more complex models by achieving minimal gains in the marginal likelihood, oscillating around the same levels of performance and requiring many steps of the search to halt as there is no penalisation term for large composite models. Discuss stopping criteria (change in BIC smaller than some threshold).

Explain that the reason that it can improve marginal likelihood by adding these irrelevant components is that multiplying by these allows the optimisation procedure to move out of the region where its stuck. Also, numerical instability due to use of approximations to the posterior may contribute to this behaviour.

4.3 Experiments with Synthetic Data

To prove that the greedy structure search procedure is able to extract structure from data, the algorithm was first applied to data drawn from a single GP prior. If the greedy search based on any of the three information criteria has potential, the procedure should be able to recover the original kernel given enough synthetic data.

The amount of data available to the procedure, as well as the signal-to-noise ratio in the data were varied across experiments. As we decrease the noise levels and add more data points, the structure search should get closer to the underlying truth, that is the original kernel used to generate the data. Signal to noise ratio values of 100 and 1 correspond to white noise of magnitudes equal to 10% and 100% of the signal magnitude. The search results shown below used *BIC light* as the guiding criterion: the correctly identified kernel structures are shown in bold.

True Kernel	N	Kernel recovered (SNR = 1)	Kernel recovered (SNR = 100)
[1] *3 dimensions*	100	**[1]**	**[1]**
	300	**[1]**	**[1]**
	500	**[1]**	**[1]**
[2] + [2] + [2] *3 dimensions*	100	[2]	[2]
	300	[2]	[2] + [2]
	500	[2]	[2] + [2]
[2 × 3] *3 dimensions*	100	**[2 × 3]**	**[2 × 3]**
	300	**[2 × 3]**	**[2 × 3]**
	500	**[2 × 3]**	**[2 × 3]**
[1] + [2 × 3] + [4] *4 dimensions*	100	[1] + [4]	[1] + [2] + [4]
	300	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
	500	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
[1] + [2 × 3] + [4] *10 dimensions*	100	[1] + [4]	[1] + [2] + [4]
	300	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
	500	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
[1] + [2 × 3] + [4] + [5 × 6] *10 dimensions*	100	[4] + [5]	[1 × 9] + [4]
	300	[1] + [2] + [4] + [5 × 6 × 7]	[1] + [1 × 5 × 6] + [2 × 3] + [4]
	500	[1] + [1 × 4] + [2 × 3] + [5 × 6]	[1] + [1 × 4 × 5 × 6] + [2 × 3] + [4]
[3 × 5 × 7] *10 dimensions*	100	[4 × 7]	**[3 × 5 × 7]**
	300	[2 × 3 × 5 × 7]	**[3 × 5 × 7]**
	500	[2 × 3 × 5 × 7]	**[3 × 5 × 7]**
[1] + [3 × 5 × 7] + [10] *10 dimensions*	100	[1 × 3] + [10]	[1 × 10]
	300	[1] + [10]	[1] + [1 × 10] + [3 × 5 × 7]
	500	[1] + [10]	**[1] + [3 × 5 × 7] + [10]**
[3 × 5 × 7 × 9] *10 dimensions*	100	[3 × 7]	[1] + [7 × 9]
	300	[7 × 9]	**[3 × 5 × 7 × 9]**
	500	**[3 × 5 × 7 × 9]**	**[3 × 5 × 7 × 9]**
[1] + [3 × 5 × 7 × 9] + [10] *10 dimensions*	100	[9 + 10]	[10]
	300	[1] + [10]	[1 × 5] + [7] + [10]
	500	[1] + [3 × 5 × 9] + [10]	**[1] + [3 × 5 × 7 × 9] + [10]**

Kernel optimal rates give a bound on the performance of the structure search: on average, the search is (by design) unable to construct kernels that have better predictive performance than the one used to generate the data. The function optimal rates give a theoretical limit on what any model could achieve, as they compare the actual function (its values before adding noise) to the data after the noise is added. As a result, kernel rates are always below the function rates. The models constructed can achieve higher accuracy than either of these by chance, but are *on average* inferior to both the kernel and the function optimal rates.

		SNR = 1			SNR = 100		
True Kernel	N	Test Accuracy	Kernel	Function	Test Accuracy	Kernel	Function
	100	**81.0%**	80.0%		**98.0**%	97.0%	
[1]	300	**82.7**%	82.0%	83.6%	**99.3%**	98.7%	99.0%
3 dimensions	500	83.6%	83.8%		**98.4%**	98.4%	
	100	74.0%	75.0%		**98.0%**	98.0%	
[2] + [2] + [2]	300	**77.3%**	77.0%	78.2%	**98.7%**	98.7%	98.8 %
3 dimensions	500	**78.2%**	78.2%		**98.6%**	98.6%	
	100	67.0%	68.0%		**89.0%**	89.0%	
[2 × 3]	300	69.7%	70.3%	74.8%	**94.7%**	91.3%	98.0 %
3 dimensions	500	71.4%	72.6%		**95.8%**	93.4%	
	100	71.0%	80.0%		82.0%	87.0%	
[1] + [2 × 3] + [4]	300	73.0%	76.3%	76.4%	**94.7%**	91.7%	97.4%
3 dimensions	500	73.4%	74.8%		**94.2%**	91.8%	
	100	71.0%	80.0%		82.0%	87.0%	
[1] + [2 × 3] + [4]	300	73.0%	76.3%	76.4%	**94.7%**	91.7%	97.4%
10 dimensions	500	73.4%	74.8%		**94.2%**	91.8%	
	100	**61.0%**	60.0%		70.0%	84.0%	
[1] + [2 × 3] + [4] + [5 × 6]	300	67.7%	70.7%	76.6%	91.0%	92.3%	97.2%
10 dimensions	500	**73.4%**	73.0%		**94.0%**	92.0%	
	100	53.0%	64.0%		**87.0%**	83.0%	
[3 × 5 × 7]	300	69.3%	71.0%	79.0%	**92.7%**	89.0%	98.4%
10 dimensions	500	73.2%	73.6%		**93.6%**	91.6%	
	100	**71.0%**	70.0%		87.0%	92.0%	
[1] + [3 × 5 × 7] + [10]	300	**73.0%**	71.3%	75.0%	**94.3%**	94.3%	97.6%
10 dimensions	500	70.0%	72.2%		**93.8%**	92.8%	
	100	57.0%	59.0%		65.0%	77.0%	
[3 × 5 × 7 × 9]	300	59.3%	70.3%	77.0%	**83.7%**	82.7%	97.0 %
10 dimensions	500	**71.0%**	69.0%		**85.6%**	82.6%	
	100	68.0%	74.0%		77.0%	81.0%	
[1] + [3 × 5 × 7 × 9] + [10]	300	70.7%	73.3%	78.2%	81.7%	89.0%	96.8%
10 dimensions	500	72.0%	72.4%		**90.6%**	89.8%	

Commentary on these results.

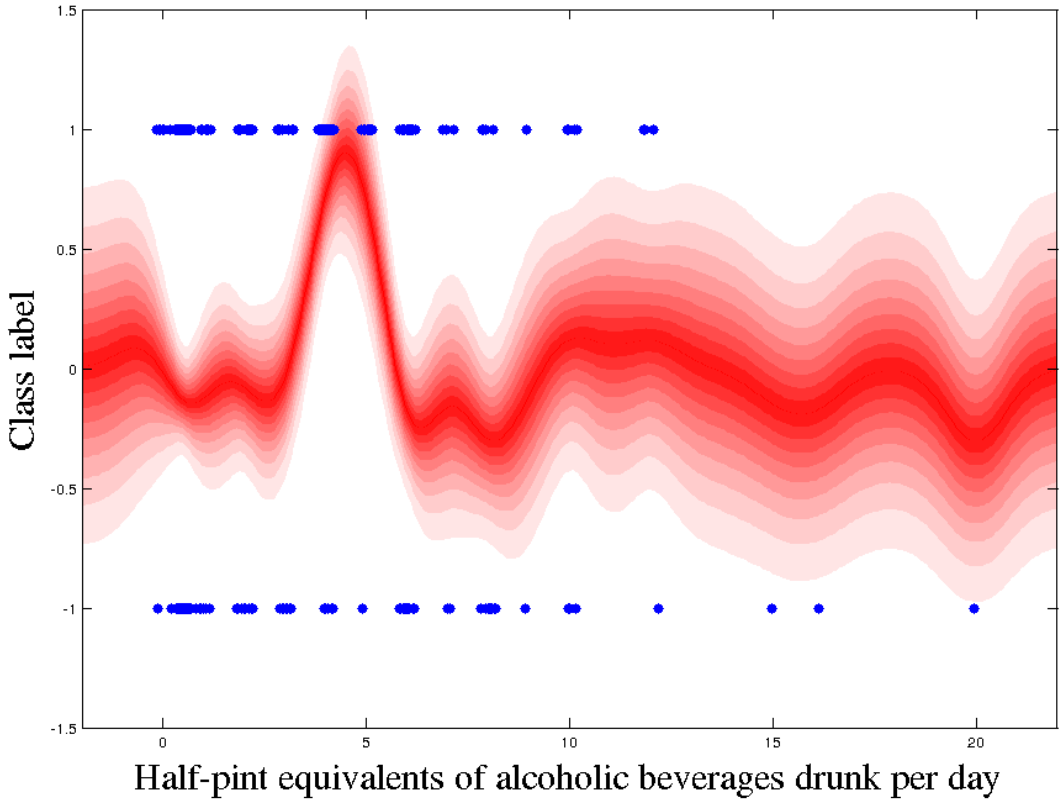
4.4 Structure Discovery for Real-world Data Sets

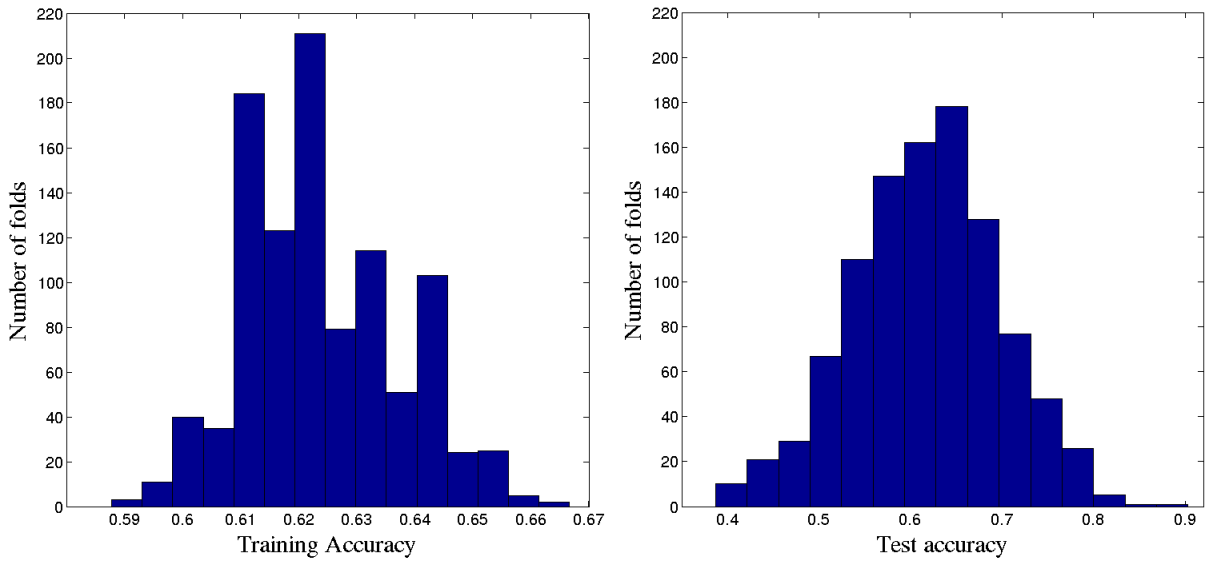
Describe the data sets first? Table of name, dimensionality, sample size.

4.4.1 Overfitting

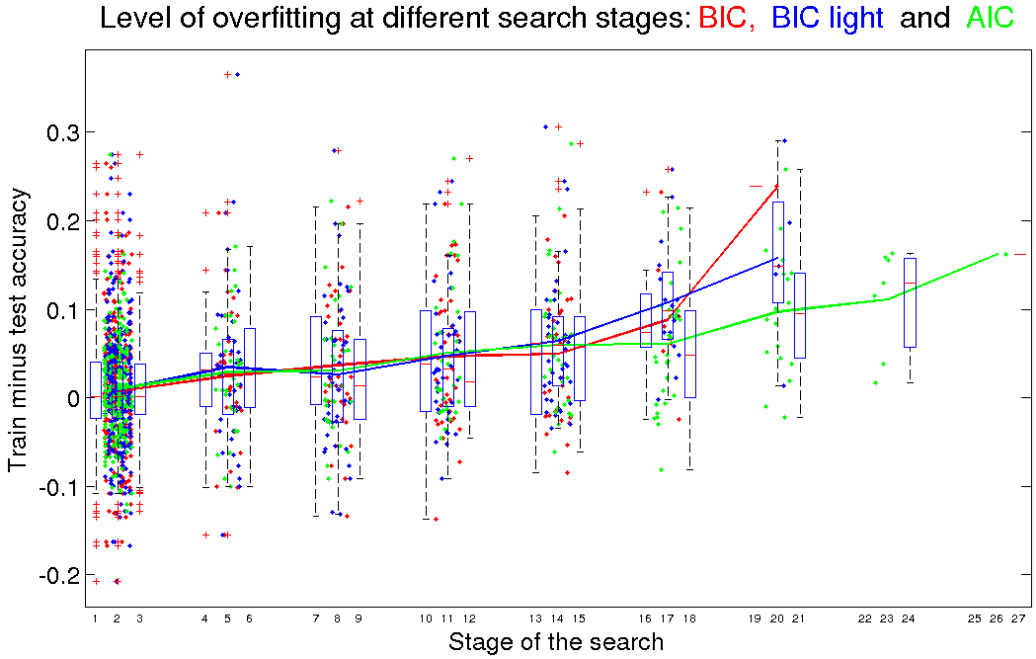
Dealing with very small and very large hyperparameters. Discuss what large ones mean (inconsequential for classification boundary) and what small ones do (overfitting). Explain how we set the minimal allowed lengthscale for each dimension and how discretised data presents a challenge.

Overfitting with large lengthscales is to be expected, so we plot the histogram of accuracies for training and testing data when doing ten fold cross validation (with many different splits, i.e. seeds). From the plots below, we see that testing accuracy is expected to vary when doing ten-fold cross validation with this amount of data. With 100 different splits (that is 1000 different data points), the training accuracy ranges from 59% to 66%, while the test accuracy varies from 40% to 90%, showing that the overfitting (or surprisingly good performance we encounter on some folds) is a something to be expected.





The number of search steps figure that shows the level of overfitting (defined as the difference between training and testing accuracy) of the three metrics at different search stages across all the experiments on real-world data (discussed in later parts of the chapter) is shown below:



4.4.2 Predictive Performance

In this section, we compare the performance of models constructed using our algorithm with related methods and show that the performance of our structurally simpler models is on par with more complicated models such as Additive GPs and Hierarchical Kernel Learning.

We also compare the performance of structure search using different information criteria (BIC, AIC, BIClight), as well as the search guided by cross-validated training accuracy. We also show the performance of the model constructed using the alternative likelihood function.

In addition to the structure search, we show the performance of the random forest method, which constructs 1000 decision trees using the training data and then uses the mode of the classifications produced by these trees to label the test set data. This method was intended to be a *ceiling* performance for our methods, as its focus is just predictive performance: it does not contribute to interpretability or our understanding of the data set considered.

The table below contains the mean classication error across 10 train-test splits between different methods. The model with the best performance is shown in bold, together with all the other models whose predictions were not significantly different from those made by the best model. Statistical significance was determined using the *paired t-test*.

Table 4.1: Classification Percent Error

Method	breast	pima	liver	heart
Logistic Regression	7.611	24.392	45.060	***16.082***
GP GAM	***5.189***	***22.419***	***29.842***	***16.839***
HKL	***5.377***	24.261	***27.270***	***18.975***
GP Squared-exp	***4.734***	***23.722***	***31.237***	***20.642***
GP Additive	***5.566***	***23.076***	***30.060***	***18.496***
GPSS (AIC)	6.430	**22.529**	28.924	19.860
GPSS (BIC)	5.980	23.440	37.010	**18.150**
GPSS (BIC light)	6.430	**22.270**	**27.500**	**17.820**
GPSS (crossValGuide)	**5.090**	23.700	-	**17.160**
Random Forest	**4.220**	**23.440**	**24.030**	**17.130**

The paired t-test is tolerant of complete outliers: if a model is on par with the best model on 9 out of 10 folds and it performs badly on the 10th fold, this case is treated as an outlier and discarded. This is why the BIC guided algorithm is deemed worse than Random Forest on the Pima dataset in spite of having the same

average error. BIC is consistently worse than BIC light (the best model) whereas Random Forest manages to outperform BIC light on some of the folds.

Chapter 5

Advancing the Automated Statistician

5.1 The Alternative Likelihood Function

5.1.1 Dealing with Outliers

The reason that GPs out of the box are easily outperformed by classification specific schemes such as SVMs is that soft-margin SVMs are less certain of their predictions, and hence are less certain about points far away from the high density regions of the data. SVMs are never more certain than the estimated level of pepper noise (outliers in the sense of noise in the target labels of the data) whereas GPs would be very certain about points relatively close to the high density clusters of one of the classes.

As a potential solution, we might want to use an alternative noise model: a mixture of likelihood functions which incorporate the presence of complete outliers into the likelihood function. The 'salt and pepper' noise level α which determines the proportion of the outliers must be learned by again maximising marginal likelihood. The probabilistic predictions of the likelihood function learned no longer range from 0 to 1, but from α to $(1 - \alpha)$, α.

Likelihood mixtures have been implemented in GPML. To determine whether this idea has potential, we can evaluate the search using the synthetic data used to evaluate the models which (so far) used the cumulative Gaussian likelihood function. The only difference is that we must add additional salt and pepper noise (random outliers) into the synthetic data to determine whether the mixture models estimate

the noise levels correctly, and to determine how much (and if) the mixture models improve performance on these datasets compared to the models which use the standard cumulative Gaussian noise model.

Include the plot of the saturating likelihood function.

What would be a good way to summarize the results, given that we currently have 8 different tables for 4 different kernels with BIC light and likMix?

5.1.2 Adding Salt and Pepper Noise

We validated our method's ability to recover known structure on a set of synthetic datasets. For several composite kernel expressions, we constructed synthetic data by first sampling 100, 300 and 500 points uniformly at random, then sampling function values at those points from a GP prior. We then added i.i.d. Gaussian noise to the functions, at various signal-to-noise ratios (SNR), as well as different amounts of salt and pepper noise (random outliers in the data set).

Table 5.1 lists the true kernels we used to generate the data. Subscripts indicate which dimension each kernel was applied to. Subsequent columns show the dimensionality D of the input space, and the kernels chosen by our search for different SNRs and different amounts of added salt and pepper noise. We also show the kernel optimal rates (the accuracy the kernel used to generate the data achieves on the noisy test set) and the function optimal rates (the rate a classifier which knew the *exact* funtion used to generate the data achieves on the noisy test data set).

The use of this likelihood function introduces two new hyperparameters which set the rates of the two likelihood functions in the mixture. As there are only two likelihood functions in the mixture, we have introduced only one new effective hyperparameter, the ratio of the two likelihood functions' contributions. If the structure search which biolds models using this likelihood function has the potential to deal with the outliers more effectively, it should be able to estimate the levels of the salt and pepper noise introduced in the synthetic data sets.

The ratios inferred sometimes estimate the level of noise correctly, but usually fail to acknowledge any noise or completely exaggerate the noise levels. This might be caused by a bug in GPML or a more fundamental problem / lack of thought put into this idea.

Table 5.1: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_2 + \mathrm{SE}_3$, $D = 3$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate
100	100	0%	$1 + 1 \times 3 + 2$	87.0%	91.0%	97.4%
300	100	0%	$1 + 2 + 3$	94.0%	95.7%	97.4%
500	100	0%	$1 + 2 + 3$	95.8%	95.4%	97.4%
100	100	5%	$1 + 2 + 3$	77.0%	80.0%	91.6%
300	100	5%	$1 \times 3 + 2$	87.0%	85.7%	91.6%
500	100	5%	$1 \times 2 \times 3$	89.8%	89.8%	91.6%
100	100	20%	1×3	69.0%	69.0%	82.0%
300	100	20%	$1 \times 3 + 2$	75.3%	73.0%	82.0%
500	100	20%	$1 \times 3 + 2$	77.6%	74.0%	82.0%
100	1	0%	$1 + 3$	64.0%	72.0%	77.4%
300	1	0%	$1 + 3$	74.3%	75.0%	77.4%
500	1	0%	$1 + 3$	75.6%	76.6%	77.4%
100	1	5%	$1 + 3$	63.0%	63.0%	74.4%
300	1	5%	1×3	70.7%	68.3%	74.4%
500	1	5%	1×3	72.6%	72.6%	74.4%
100	1	20%	1×3	53.0%	60.0%	68.8%
300	1	20%	1×3	65.3%	65.3%	68.8%
500	1	20%	1×3	66.2%	67.8%	68.8%

Table 5.2: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_2 + \mathrm{SE}_3$, $D = 3$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate	Noise ratio
100	100	0%	1×3	92.0%	91.0%	97.4%	0.0%
300	100	0%	$1 + 2 + 3$	94.0%	95.7%	97.4%	0.0%
500	100	0%	$1 \times 3 + 2 + 3$	94.8%	95.4%	97.4%	0.0%
100	100	5%	$1 + 3$	80.0%	79.0%	91.6%	0.0%
300	100	5%	$1 + 1 \times 3 + 2$	87.0%	89.3%	91.6%	11.8%
500	100	5%	$1 \times 3 + 2 + 2 \times 2$	89.2%	90.6%	91.6%	13.9%
100	100	20%	1×3	69.0%	64.0%	82.0%	0.0%
300	100	20%	1×3	75.0%	74.7%	82.0%	32.2%
500	100	20%	$1 \times 3 + 2$	77.2%	75.0%	82.0%	33.3%
100	1	0%	$1 + 3$	64.0%	71.0%	77.4%	0.0%
300	1	0%	$1 + 3$	74.0%	73.7%	77.4%	5.3%
500	1	0%	$1 + 3$	76.0%	76.4%	77.4%	0.0%
100	1	5%	$1 + 3$	63.0%	63.0%	74.4%	0.0%
300	1	5%	$1 + 3$	72.0%	71.0%	74.4%	12.9%
500	1	5%	1×3	72.6%	73.2%	74.4%	0.0%
100	1	20%	1×3	53.0%	61.0%	68.8%	0.0%
300	1	20%	1×3	65.3%	65.7%	68.8%	0.1%
500	1	20%	1×3	68.0%	67.6%	68.8%	0.3%

Table 5.3: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_4$, $D = 4$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate
100	100	0%	$1 + 2 \times 3 + 4$	87.0%	92.0%	97.4%
300	100	0%	$1 + 2 \times 3 + 4$	94.0%	94.7%	97.4%
500	100	0%	$1 + 2 \times 3 + 4$	95.6%	96.2%	97.4%
100	100	5%	$1 + 2$	81.0%	76.0%	92.0%
300	100	5%	$1 + 2 + 3 \times 4$	85.7%	84.0%	92.0%
500	100	5%	$1 \times 4 + 2 \times 3 + 3$	87.6%	88.6%	92.0%
100	100	20%	2×4	67.0%	67.0%	82.0%
300	100	20%	$2 \times 3 + 4$	76.0%	73.7%	82.0%
500	100	20%	$2 + 3 \times 4$	77.0%	79.8%	82.0%
100	1	0%	2	68.0%	67.0%	76.0%
300	1	0%	$1 + 2 \times 3$	72.3%	70.3%	76.0%
500	1	0%	$1 + 2 \times 3$	72.2%	73.2%	76.0%
100	1	5%	2	67.0%	58.0%	72.2%
300	1	5%	1×2	71.0%	64.3%	72.2%
500	1	5%	$1 \times 2 \times 3$	70.6%	68.0%	72.2%
100	1	20%	2	59.0%	61.0%	69.0%
300	1	20%	$2 \times 3 \times 4$	65.3%	62.3%	69.0%
500	1	20%	$2 \times 3 \times 4$	64.8%	64.8%	69.0%

Table 5.4: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_4$, $D = 4$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate	Noise ratio
100	100	0%	$1 + 2 \times 3 + 4$	87.0%	92.0%	97.4%	0.0%
300	100	0%	$1 + 2 \times 3 + 4$	94.0%	94.7%	97.4%	0.0%
500	100	0%	$1 + 2 \times 3 + 4$	95.4%	96.2%	97.4%	0.0%
100	100	5%	$1 \times 2 + 2$	81.0%	78.0%	92.0%	0.0%
300	100	5%	$1 + 2 \times 3 + 4$	87.3%	88.3%	92.0%	9.9%
500	100	5%	$1 + 2 \times 3 + 3 + 4$	90.2%	89.6%	92.0%	12.8%
100	100	20%	$1 \times 4 + 2$	64.0%	65.0%	82.0%	18.8%
300	100	20%	$2 \times 3 + 4$	75.7%	74.3%	82.0%	9.4%
500	100	20%	$1 \times 3 + 2 + 4$	77.0%	78.8%	82.0%	30.5%
100	1	0%	2	69.0%	65.0%	76.0%	54.4%
300	1	0%	$1 + 2 \times 3$	72.3%	73.0%	76.0%	0.0%
500	1	0%	$1 + 2 \times 3 + 4$	72.2%	73.2%	76.0%	0.0%
100	1	5%	2	67.0%	58.0%	72.2%	43.1%
300	1	5%	1×2	71.0%	65.0%	72.2%	2.6%
500	1	5%	$1 \times 2 + 3$	70.2%	68.8%	72.2%	29.0%
100	1	20%	2	59.0%	63.0%	69.0%	40.6%
300	1	20%	2	63.3%	60.7%	69.0%	51.3%
500	1	20%	2×3	63.4%	64.2%	69.0%	57.6%

Table 5.5: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_7 + \mathrm{SE}_{10}$, $D = 10$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate
100	100	0%	$1 \times 9 + 10$	61.0%	88.0%	96.0%
300	100	0%	$1 + 1 \times 10 + 3 \times 7$	92.0%	92.7%	96.0%
500	100	0%	$1 + 1 \times 3 \times 7 \times 10 + 10$	94.2%	94.6%	96.0%
100	100	5%	$1 \times 9 + 10$	53.0%	71.0%	91.8%
300	100	5%	$1 + 3 \times 7 + 6 \times 10$	82.0%	81.3%	91.8%
500	100	5%	$1 \times 3 \times 7 \times 10 + 10$	85.0%	86.2%	91.8%
100	100	20%	1	49.0%	64.0%	79.8%
300	100	20%	$1 + 10$	60.0%	70.0%	79.8%
500	100	20%	$1 \times 3 \times 7 \times 10$	74.2%	75.2%	79.8%
100	1	0%	10	59.0%	70.0%	74.4%
300	1	0%	$1 \times 3 \times 7 \times 10 + 10$	71.3%	72.7%	74.4%
500	1	0%	$1 \times 10 + 3 \times 7 + 9$	72.0%	71.4%	74.4%
100	1	5%	10	55.0%	66.0%	71.4%
300	1	5%	1×10	58.7%	68.7%	71.4%
500	1	5%	$1 + 10$	60.6%	69.4%	71.4%
100	1	20%	3	55.0%	56.0%	65.4%
300	1	20%	10	58.0%	61.7%	65.4%
500	1	20%	1×10	58.2%	62.0%	65.4%

Table 5.6: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_7 + \mathrm{SE}_{10}$, $D = 10$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate	Noise ratio
100	100	0%	10	62.0%	88.0%	96.0%	20.4%
300	100	0%	$1 + 3 \times 7 + 10$	92.0%	93.0%	96.0%	0.0%
500	100	0%	$1 \times 7 \times 10$	75.0%	94.6%	96.0%	5.4%
100	100	5%	$1 \times 9 + 10$	53.0%	75.0%	91.8%	0.0%
300	100	5%	$1 + 3 \times 7 + 10$	84.7%	86.7%	91.8%	7.0%
500	100	5%	$1 \times 10 + 3 \times 7$	87.8%	89.2%	91.8%	11.3%
100	100	20%	1	50.0%	64.0%	79.8%	49.5%
300	100	20%	$1 + 10$	58.3%	72.7%	79.8%	55.1%
500	100	20%	$1 + 3 \times 7 + 10$	77.2%	77.4%	79.8%	28.2%
100	1	0%	10	58.0%	73.0%	74.4%	6.9%
300	1	0%	$1 \times 3 \times 7 \times 10 + 10$	71.3%	72.7%	74.4%	0.0%
500	1	0%	6	48.2%	72.4%	74.4%	90.3%
100	1	5%	10	55.0%	67.0%	71.4%	1.4%
300	1	5%	1×10	58.7%	69.0%	71.4%	0.0%
500	1	5%	10	57.2%	70.4%	71.4%	74.2%
100	1	20%	10	62.0%	57.0%	65.4%	53.0%
300	1	20%	10	58.0%	62.7%	65.4%	0.2%
500	1	20%	10	58.8%	62.8%	65.4%	0.1%

Table 5.7: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7 + \mathrm{SE}_9$, $D = 10$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate
100	100	0%	$3 \times 5 \times 7$	85.0%	86.0%	97.0%
300	100	0%	$3 \times 5 \times 7 + 9$	93.7%	93.0%	97.0%
500	100	0%	$3 \times 5 \times 7$	91.4%	92.2%	97.0%
100	100	5%	$3 \times 5 \times 7$	78.0%	76.0%	91.6%
300	100	5%	$3 \times 5 \times 7$	84.0%	83.7%	91.6%
500	100	5%	$3 \times 5 \times 7$	86.2%	83.6%	91.6%
100	100	20%	8	49.0%	59.0%	82.0%
300	100	20%	$3 \times 5 \times 7$	68.3%	66.0%	82.0%
500	100	20%	$3 \times 5 \times 7$	72.2%	66.0%	82.0%
100	1	0%	$1 \times 3 \times 4 \times 5 + 7$	59.0%	66.0%	74.2%
300	1	0%	$3 \times 5 \times 7 + 9$	71.7%	72.7%	74.2%
500	1	0%	$1 + 3 \times 5 \times 7$	73.0%	70.6%	74.2%
100	1	5%	$1 \times 3 \times 4 \times 5 + 7$	55.0%	62.0%	70.8%
300	1	5%	$3 \times 5 \times 7$	64.3%	68.7%	70.8%
500	1	5%	$3 \times 5 \times 7$	70.4%	67.4%	70.8%
100	1	20%	$3 \times 5 \times 9$	52.0%	64.0%	66.4%
300	1	20%	$3 \times 7 \times 8$	55.7%	61.7%	66.4%
500	1	20%	3×7	56.4%	62.6%	66.4%

Table 5.8: True kernel: $\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7 + \mathrm{SE}_9$, $D = 10$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate	Noise ratio
100	100	0%	$3 \times 5 \times 7$	85.0%	86.0%	97.0%	0.0%
300	100	0%	$3 \times 5 \times 7 + 9$	93.3%	93.0%	97.0%	0.0%
500	100	0%	$3 \times 5 \times 7 + 5$	92.2%	92.2%	97.0%	0.0%
100	100	5%	$3 \times 5 \times 7$	78.0%	77.0%	91.6%	0.0%
300	100	5%	$3 \times 5 \times 7$	84.0%	83.7%	91.6%	0.0%
500	100	5%	$3 \times 5 \times 7 + 5 + 10$	85.6%	84.0%	91.6%	6.2%
100	100	20%	9	45.0%	59.0%	82.0%	0.0%
300	100	20%	6	46.3%	65.3%	82.0%	77.2%
500	100	20%	$3 \times 5 \times 7$	72.2%	66.6%	82.0%	0.0%
100	1	0%	$1 \times 3 \times 4 \times 5 + 7$	59.0%	66.0%	74.2%	0.0%
300	1	0%	$1 + 10$	50.3%	72.7%	74.2%	93.8%
500	1	0%	$3 \times 5 \times 7$	72.2%	71.0%	74.2%	0.0%
100	1	5%	$1 \times 3 \times 4 \times 5 + 7$	55.0%	62.0%	70.8%	0.0%
300	1	5%	4	47.0%	68.7%	70.8%	98.4%
500	1	5%	2	48.4%	67.2%	70.8%	84.5%
100	1	20%	$2 \times 3 \times 5$	60.0%	64.0%	66.4%	0.0%
300	1	20%	$3 \times 7 \times 8$	55.7%	59.0%	66.4%	0.0%
500	1	20%	7	55.6%	60.8%	66.4%	0.0%

5.1.3 Likelihood Mixture Performance on Real World Data

Table 5.9: Classification Percent Error

Method	breast	pima	liver	heart
GPSS (AIC)	6.430	**22.529**	28.924	19.860
GPSS (BIC)	5.980	23.440	37.010	**18.150**
GPSS (BIC light)	6.430	**22.270**	**27.500**	**17.820**
GPSS (likMix)	**11.240**	**23.180**	**28.370**	**16.460**
GPSS (crossValGuide)	**5.090**	23.700	-	**17.160**
Random Forest	**4.220**	**23.440**	**24.030**	**17.130**

5.2 Bayesian Model Averaging

Our implementation averages class labels across different models, and α shows the BIC values scaling factor used to assign the weight factors to different classes. From the set of models in the layers above, in and below the current model, we use their BIC values $b_1, b_2, \ldots$ to assign each model a weight:

$$w_i = \frac{e^{-\alpha b_i}}{\sum_j e^{-\alpha b_j}}$$

And we then average the predictions of all of these models to determine the class label of each test point. The differences in performance across all folds for the four real-world data sets evaluated are shown in the tables below:

Table 5.10: Bayesian Model Averaging, classification errors across folds.

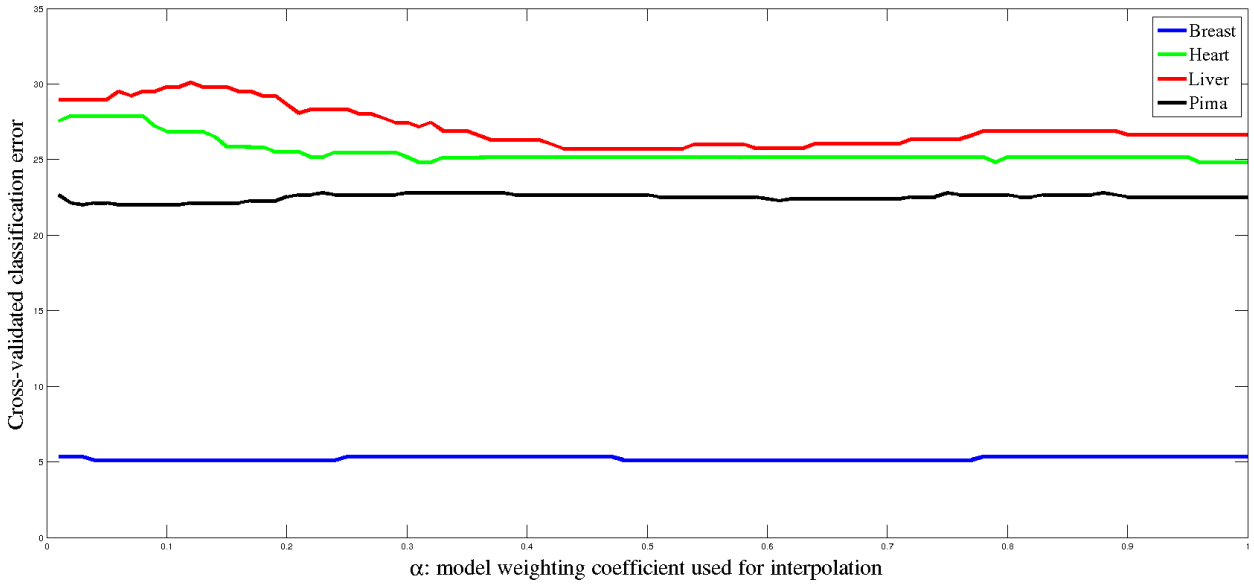
Liver		
Best model	$\alpha = 0.5$	$\alpha = 1$
23.53%	**20.59%**	**20.59%**
20.00%	22.86%	20.00%
37.14%	37.14%	37.14%
40.00%	**31.43%**	**34.29%**
28.57%	31.43%	28.57%
26.47%	**23.53%**	26.47%
23.53%	**20.59%**	26.47%
26.47%	**23.53%**	26.47%
25.71%	28.57%	28.57%
23.53%	**17.65%**	**17.65%**
27.50%	**25.73%**	**26.62%**

Pima		
Best model	$\alpha = 0.1$	$\alpha = 1$
21.05%	22.37%	21.05%
23.38%	23.38%	24.68%
15.58%	19.48%	16.88%
18.18%	19.48%	18.18%
19.48%	20.78%	22.08%
29.87%	**28.57%**	**28.57%**
29.87%	29.87%	36.36%
19.48%	**16.88%**	**14.29%**
25.00%	**19.74%**	**23.68%**
20.78%	**19.48%**	**19.48%**
22.27%	**22.00%**	22.53%

Heart		
Best model	$\alpha = 0.5$	$\alpha = 1$
17.24%	20.69%	17.24%
13.33%	53.33%	53.33%
20.69%	20.69%	24.14%
6.90%	**3.45%**	**3.45%**
16.67%	**10.00%**	**10.00%**
26.67%	26.67%	30.00%
30.00%	53.33%	53.33%
20.00%	23.33%	23.33%
16.67%	20.00%	20.00%
10.00%	16.67%	16.67%
17.82%	24.82%	25.15%

Breast		
Best model	$\alpha = 0.5$	$\alpha = 1$
2.27%	2.27%	2.27%
4.55%	4.55%	4.55%
11.11%	**8.89%**	**8.89%**
15.56%	**11.11%**	**11.11%**
2.27%	2.27%	2.27%
10.87%	**6.52%**	**6.52%**
0.00%	0.00%	0.00%
2.22%	2.22%	2.22%
4.35%	4.35%	4.35%
11.11%	**8.89%**	11.11%
6.43%	**5.11%**	**5.33%**

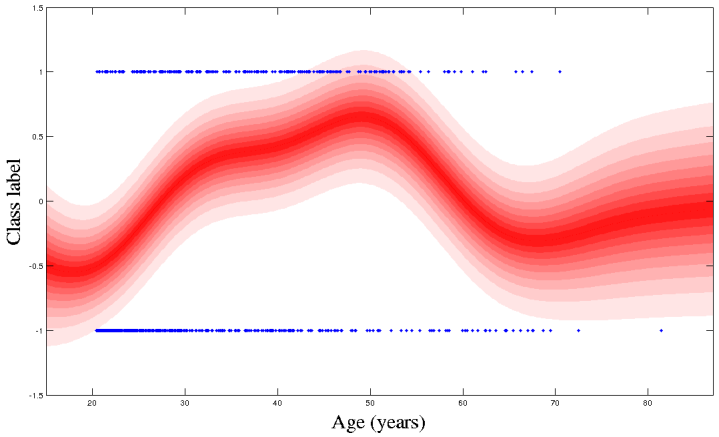
Estimating a good value for the scaling factor α is not straightforward, but its values are not as important for subsequent test performance as one might expect. The dependence of the cross-validated test error on its value is shown in the figure below:

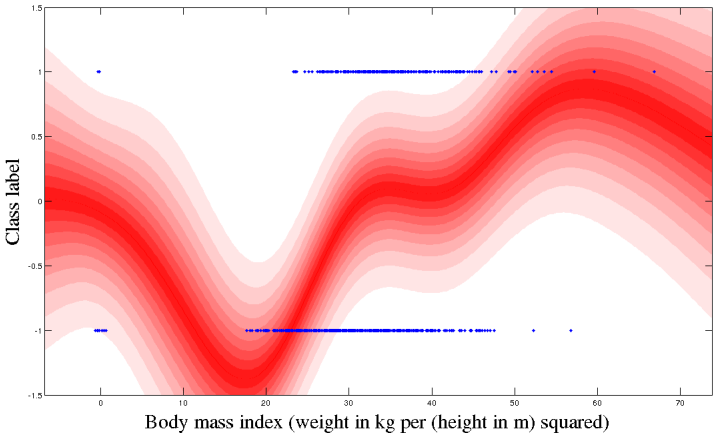


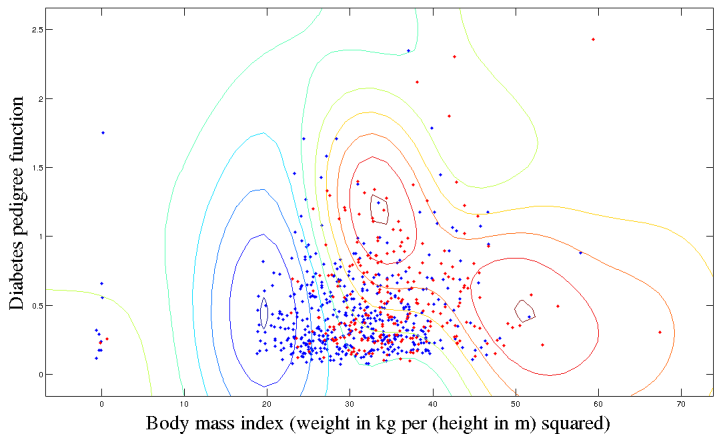
5.3 Providing Interpretability

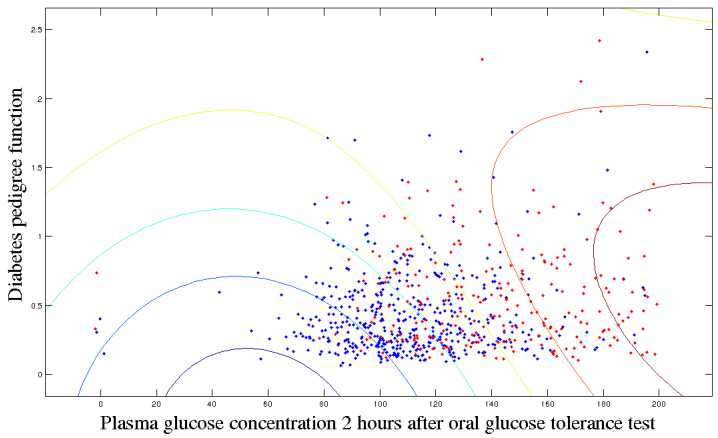
5.3.1 Pima

The Pima Indian Diabetes dataset contains medical measurements of 768 individuals, together with an indicator of whether they later developed diabetes or not. There are 8 attributes in total, 4 of which feature in the kernel structures returned by the kernel discovery procedure. These are dimensions 2, 6, 7 and 8: plasma glucose concentration, body mass index (BMI), diabetes pedigree function and the age of the patient. The remaining four attributes seem to carry less information regarding the likelihood of developing diabetes, at least in a dataset of this size and using this search procedure. These four features are the number of pregnancies, diastolic blood pressure, triceps skin fold thickness and 2-hour serum insulin measures. The structure returned most often (4 times using 10-fold cross validation for computing classification accuracy) was $[2 \times 7] + [6] + [8]$, closely followed by $[2] + [8] + [6 \times 7]$, which occured for 3 folds. The four posterior plots indicating the class boundaries determined with respect to these dimensions are shown below.

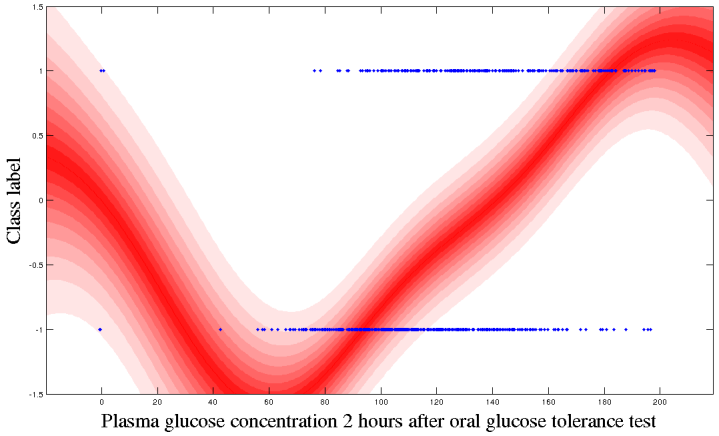


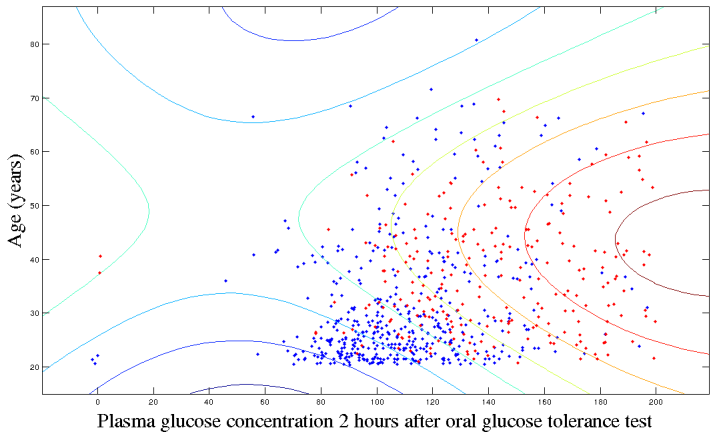


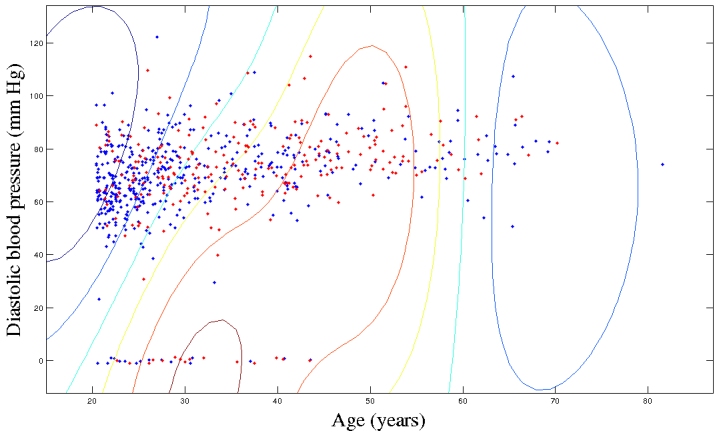


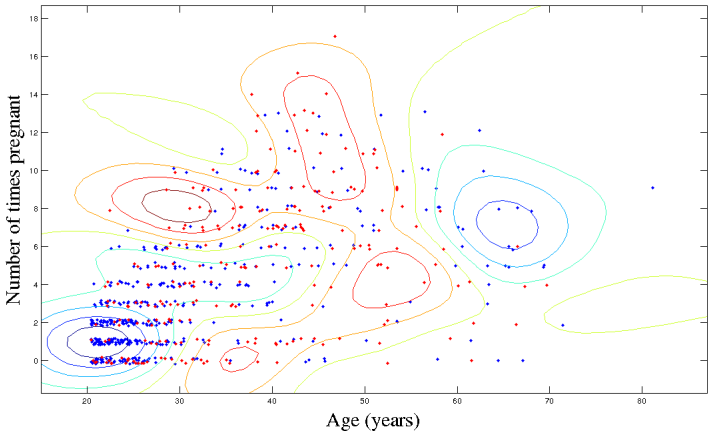


In addition to these additive components, the remaining three folds' structures contained occurances of [2], [2 × 3 × 7] and [2 × 3 × 7]. Showing the three-way dependencies in 2D is not straightforward. Below, we show the posterior plots for [2], [2 × 8] and [3 × 8], showing that interactions of these dimensions produce points well separated by the classifiers produced (with the possible exception of the interaction between age and diastolic blood pressure). The two way interaction between the number of pregnancies and age is also shown: this dependency was found during various stages of the search but discarded, even though the classifier seems to observe the general pattern present in the data set.









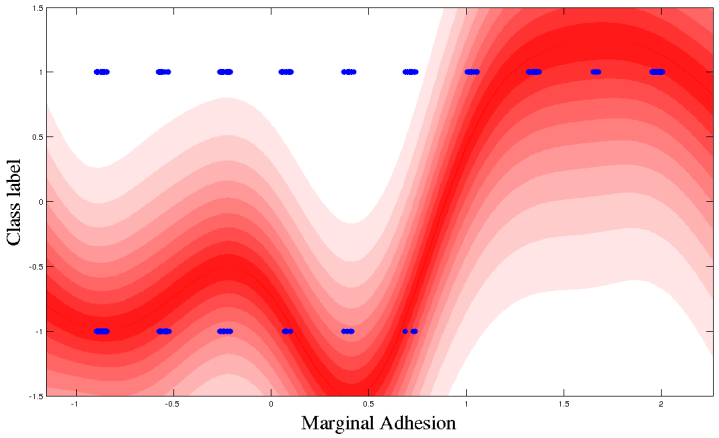
5.3.2 Breast

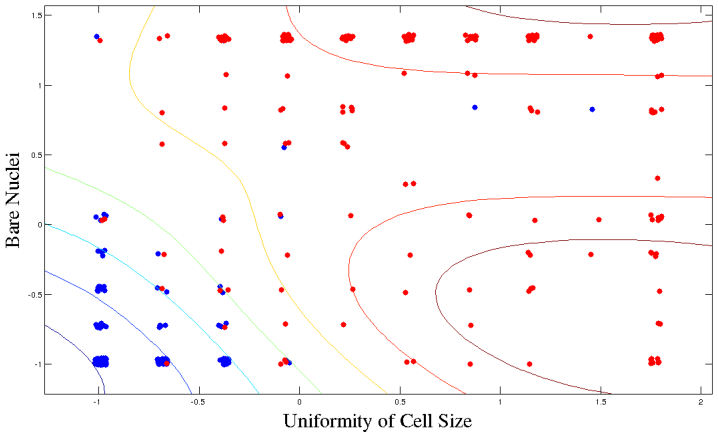
Wisconsin Diagnostic Breast Cancer Dataset is a 9-dimensional dataset which contains measured attributes of 449 individuals, together with an indicator of whether or not they later developed breast cancer. Across the 10 folds generated, there were nine different kernel structures that the procedure returned, with seven out of the nine attributes utilised in these structures. The dimensions most frequently found were 1, 2, 4, 6 and 8. Most kernels returned represent some combination of

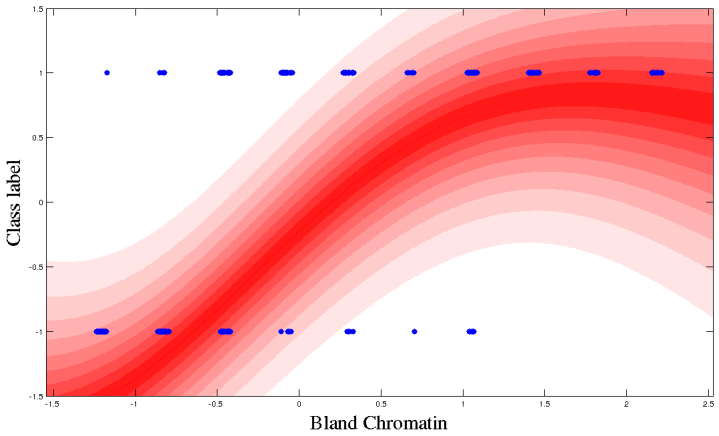
these five base kernels. Dimension 5 features once, and dimension 7 three times. Interestingly, whenever SE_7 was introduced, the accuracy increased far above the levels attained in the other kernels (ranging from 97% to 100%), but it was part of the final kernel in only three out of the ten folds evaluated.

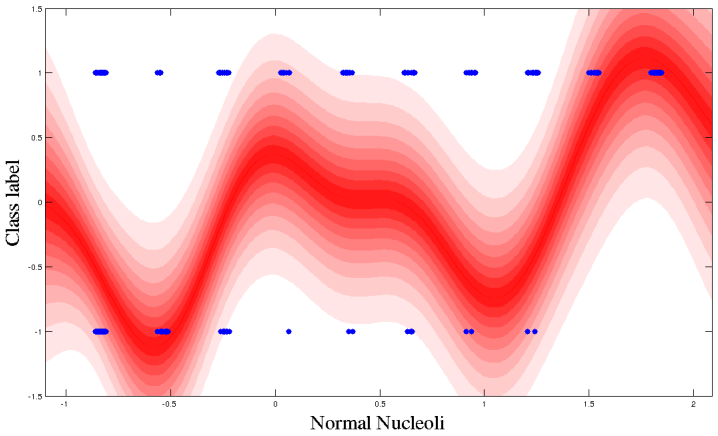
Below, we show those components of the 10 final structures which represent 1 or 2-way interactions. $[1\times2\times6]$ features in three folds, as do some other high dimensional interactions, but we are still unable to visualise those.

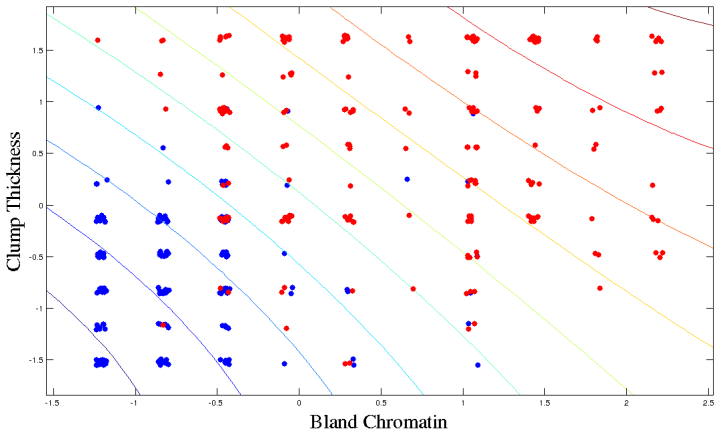
The first six plots show individual components of the final kernel structures obtained using the structure search (not necessarily across the same fold). The final row shows dimensions 1 and 5 using a single SE kernel in the respective dimension.

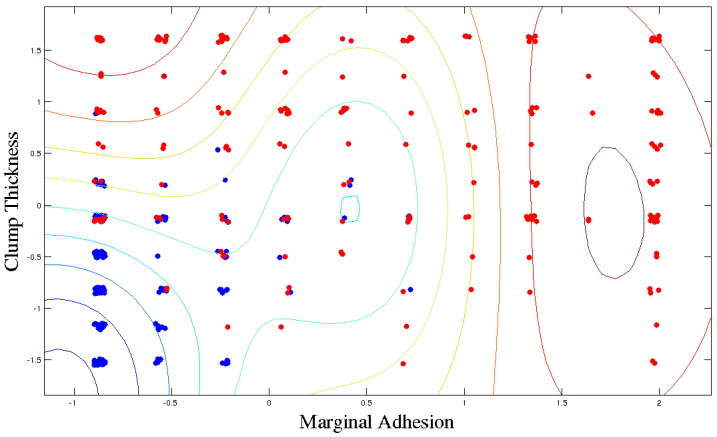


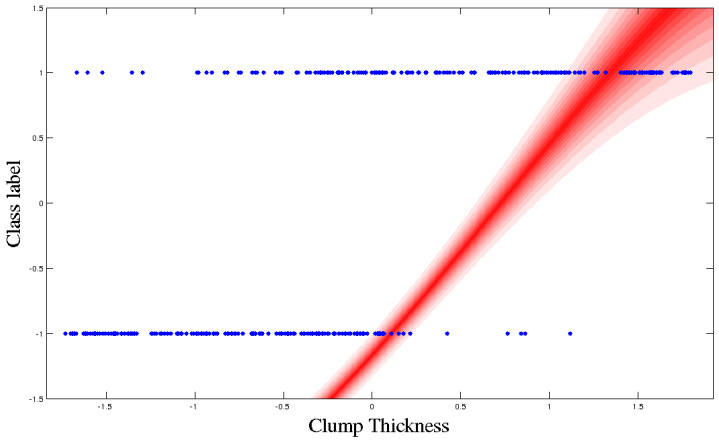


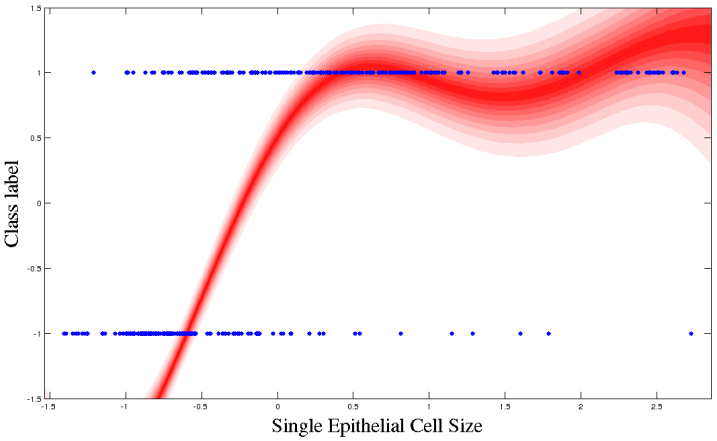





Clump Thickness
Bland Chromatin


Clump Thickness
Marginal Adhesion


Class label
Clump Thickness


Class label
Single Epithelial Cell Size

Chapter 6

Summary and Conclusions

6.1 Further Work

Bibliography

[1] Zoubin Ghahramani. Bayesian nonparametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society*, 2012.

[2] Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. *Advances in Neural Information Processing Systems 13*, 2001.

[3] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA, 2006.

[4] Peter Orbanz and Yee Whye Teh. Bayesian nonparametric models. *Encyclopedia of Machine Learning*, 2010.

[5] Carl Edward Rasmussen. Gaussian processes in machine learning. *Lecture Notes in Computer Science (LNCS), volume 3176, pages 63-71*, 2004.

[6] Roger Grosse Joshua B. Tenenbaum David Duvenaud, James Robert Lloyd and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *30th International Conference on Machine Learning, Atlanta, Georgia, USA*, June 2013.

[7] Malte Kuss and Carl Edward Rasmussen. Assessing approximations for gaussian process classification. *Advances in Neural Information Processing Systems 18, pages 699-706*, April 2006.

[8] Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary gaussian process classification. *Journal of Machine Learning Research*, 2005.

[9] M. Christoudias, R. Urtasun, and T. Darrell. Bayesian localized multiple kernel learning. *Technical report, EECS Department, University of California, Berkeley*, 2009.

[10] N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005.

[11] R. Salakhutdinov and G. Hinton. Using deep belief nets to learn covariance kernels for Gaussian processes. *Advances in Neural information processing systems*, 20:1249–1256, 2008.

[12] D. Duvenaud, H. Nickisch, and C.E. Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems*, 2011.

[13] Roger Grosse Joshua B. Tenenbaum James Robert Lloyd, David Duvenaud and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. *Association for the Advancement of Artificial Intelligence (AAAI)*, July 2014.

[14] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.