The Automated Statistician for Gaussian Process Classification

Nikola Mrkšić
Trinity College

UNIVERSITY OF CAMBRIDGE

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

May 8, 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: 0

Signed:

Date:

Abstract

Contents

Chapter 1

Introduction

1.1 Building an Automated Statistician

1.2 Contribution

Built a system for automatic kernel discovery for Gaussian Process classification...

Chapter 2

Gaussian Processes

0. Describe Bayesian paradigm and Bayes rule, and how this leads into wanting to have a prior over functions...

1. Define GPs, explain how it is fully specified by the mean and covariance function... Discuss covariance functions. Talk about function as infinite vector to generalize from multinomial Gaussian to this.

2. GP as a prior on function, show some function draws from the prior...

3. Marginal likelihood as criterion for model selection and fitting the hyper-parameters. write out nlml formula and discuss that the first term penalises poor fit, second model complexity, third is normalization. Say this is type II ML.

2.1 Kernels for Gaussian Processes

Describe different types of kernels.

2.2 Gaussian Process Classification

Explain why it is harder, non-Gaussian likelihood...

2.2.1 The Laplace Approximation to Marginal Likelihood

2.2.2 Expectation Propagation

2.2.3 Variational Bayes

Chapter 3

Related Work

3.1 Kernel learning

General review of work done in this area, with different models.

3.2 Kernel structure discovery for regression

Unsupervised structure discovery - mention that paper, maybe initially, as similar work...

Additive GPs....

ng a critique of related work, and describing what is insufficient (and how you do better!)

Discuss different information criteria, i.e. the fact that RQs can be penalised more than an SE... preferring products to additive components, say that this is something we evaluate thoroughly and try to improve on.

Chapter 4

Kernel Structure Discovery for GP Classification

Write an intro to doing this for classification, why it is harder apart from the non-Gaussian likelihoods (less information than regression, harder interpretability).

4.1 Defining the Kernel Grammar

Draw the basic kernels, describe additive and product kernels. Compare with the regression case, explain why the RQ and Periodic kernel make less sense.

Present challenges for classification, difference from regression, lack of clear component interpretability, as opposed to i.e. time-series data.

4.1.1 The Search Operators

Adding or multiplying with a base kernel. Discuss what adding two kernels or multiplying them accomplishes.

4.2 Model selection

Discuss the use of marginal likelihood as the criterion for model selsection. Explain that the hyperparameters are found using conjugate descent, in order to minimise marginal likelihood. Bayesian Occam's razor.

4.3 Optimising the Hyperparameters

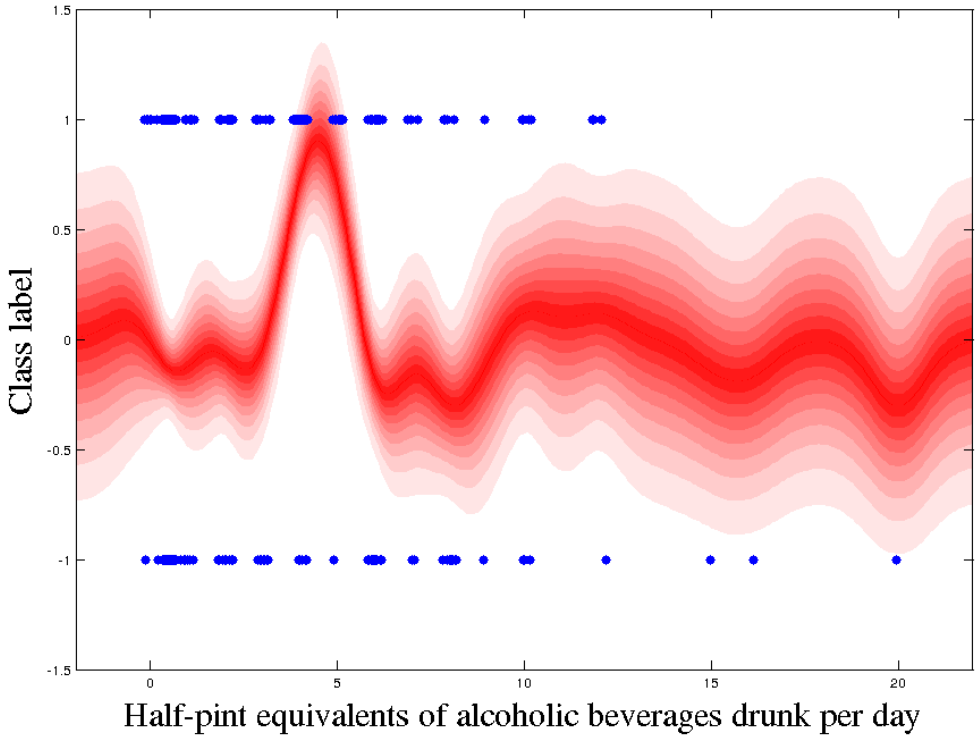
4.3.1 Conjugate descent

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 always converges to the same hyperparameters as the minimization that uses the entire data set for the whole process.

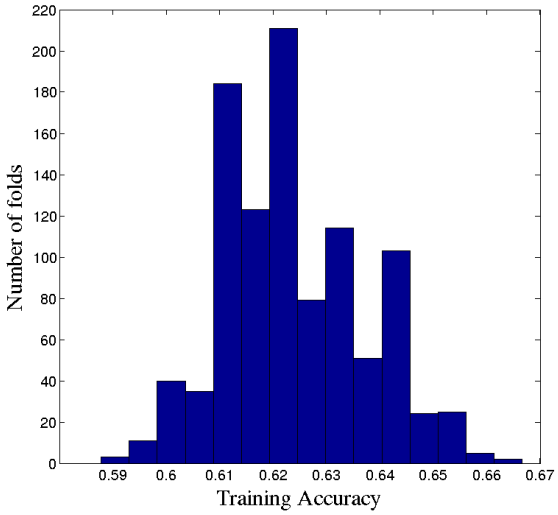
Discuss parallelisation and the use of the cluster. Discuss and add a table of running times and numbers of restarts performed.

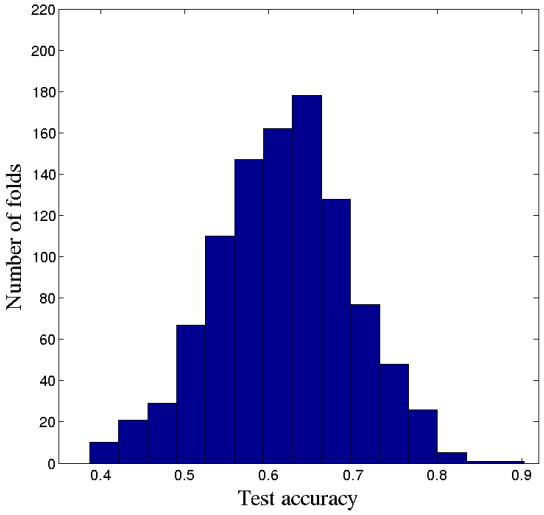
4.3.2 Overfitting

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.


Number of folds
Training Accuracy
0
20
40
60
80
100
120
140
160
180
200
220
0.59
0.6
0.61
0.62
0.63
0.64
0.65
0.66
0.67


Number of folds
Test accuracy
0
20
40
60
80
100
120
140
160
180
200
220
0.4
0.5
0.6
0.7
0.8
0.9

4.4 Guiding the Structure Search

Discuss the greedy search and backtracking, as well as potential feature reduction. To be tested on ionosphere (32 dimensional). Perhaps compare performance with and without using feature selection?

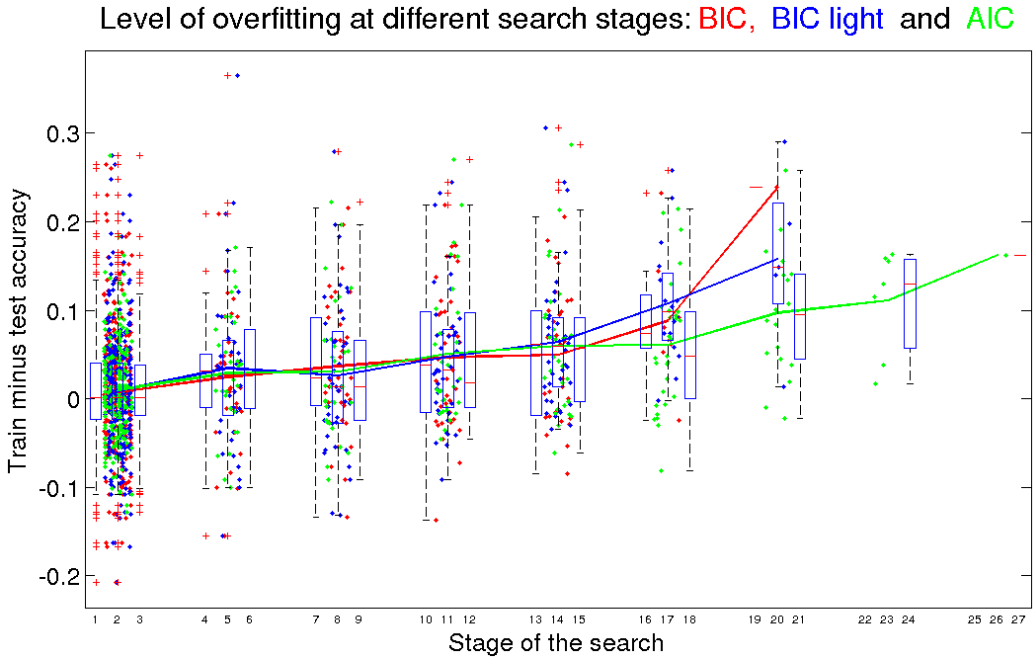
4.4.1 Bayesian and Akaike Information Criteria

Definitions and examples of successful applications.

4.4.2 The Number of Effective Hyperparameters

Outline BIC light, present it as middle ground between full BIC and AIC. Discuss the (lack of) significance of signal variance to the classification boundary.

The number of search steps figure that shows overfitting across the three metrics is shown below:



4.4.3 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.

4.5 Adapting the likelihood function

4.5.1 Dealing with Outliers

a problem with classification and BIC 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 length-scales are the most important factor (as they determine the predictive mean and where the crossings exactly are). Larger lengthscales equate to multiplication with a constant function, so less beneficial. Signal variances are useful for computing the predictive variances, but we only care about the predictive means for determining the target label. Not counting signal frequencies might add a quick boost to quality of BIC as a model discriminator (i.e. not number of product terms, but 0 - wont decrease by 2, but might help our choice-making).

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 close to one of the clusters...

As a potential solution, we might want to use a @likMix which adds the pepper noise into the likelihood function, determining the level of the outliers and thus having the likelihood (i.e. sigmoid) which doesnt range from 0 to 1, but from α to $(1-\alpha)$, where α is the level of pepper noise, which is, again, to be learnt from the data? This can be implemented in GPML using @likMix, and is something we can first evaluate w.r.t. synthetic data we previously generated. The only difference is that now we add additional salt and pepper noise (random outliers) into the data and we compare performance (on this dataset) of likMix vs likErf on its own.

4.6 Bayesian Model Averaging

Averaging assigned labels or probabilities converges to the same thing when many models are used.

A plot showing how the accuracy changes with different α coefficients.

The current 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:

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

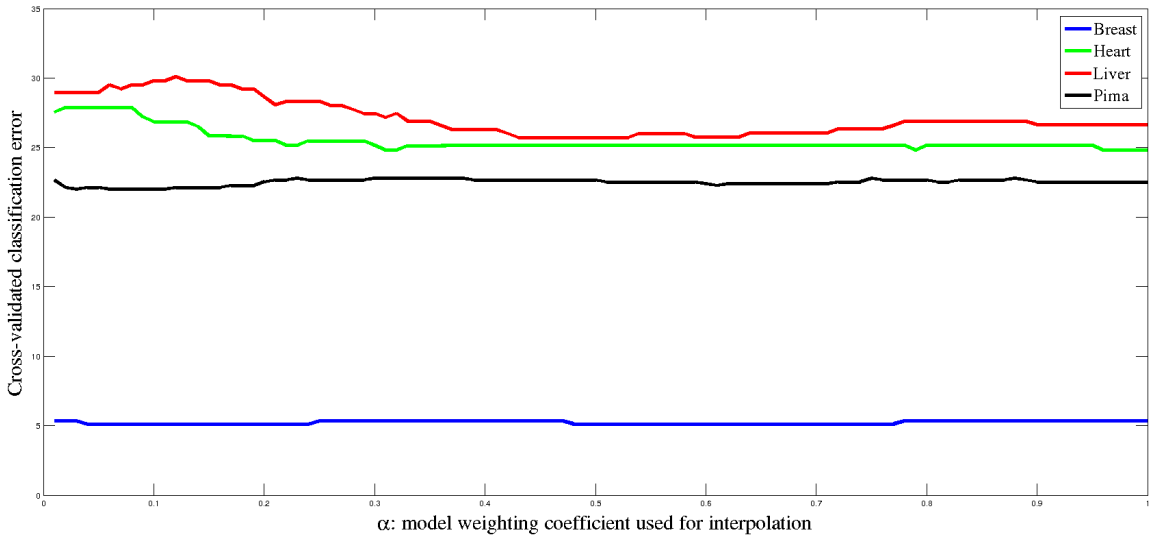
Table 4.1: Bayesian Model Averaging, classification errors across folds.

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%

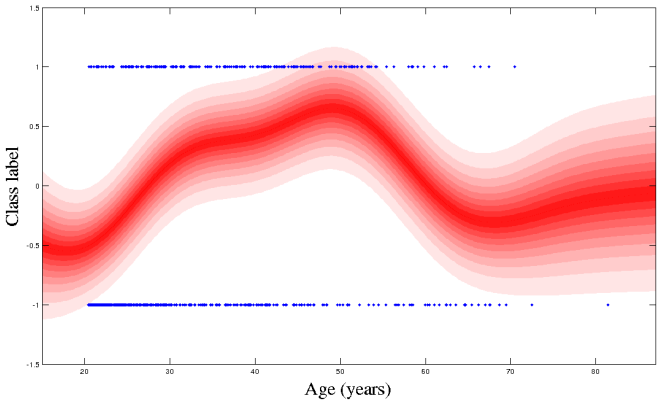
Breast		
Best model	$\alpha = 0.5$	$\alpha = 1$
2.27%	2.27%	2.27%
4.55%	4.55%	4.55%
11.11%	**8.89%**	**8.89%**
15.56%	**11.11%**	**11.11%**
2.27%	2.27%	2.27%
10.87%	**6.52%**	**6.52%**
0.00%	0.00%	0.00%
2.22%	2.22%	2.22%
4.35%	4.35%	4.35%
11.11%	**8.89%**	11.11%

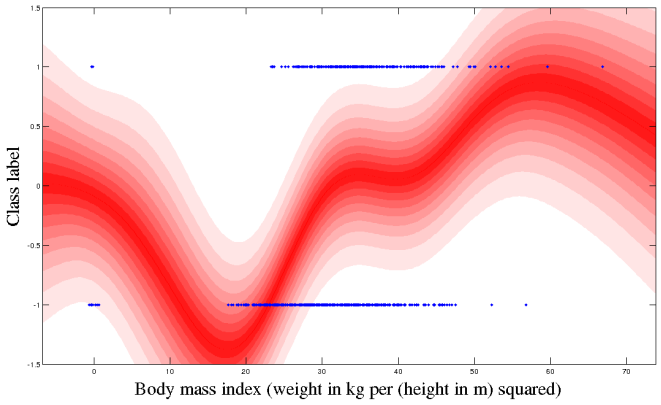


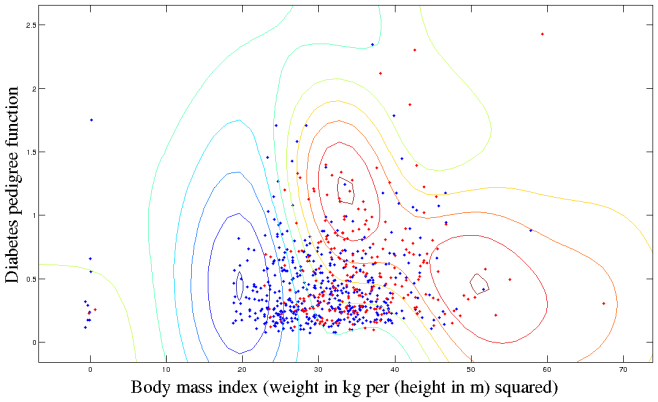
4.7 Providing Interpretability

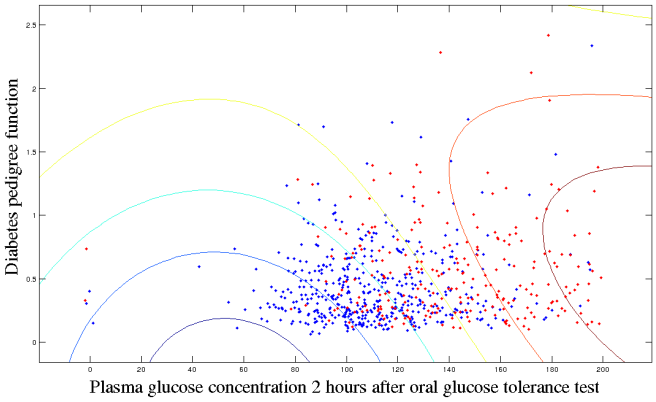
4.7.1 Pima

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.

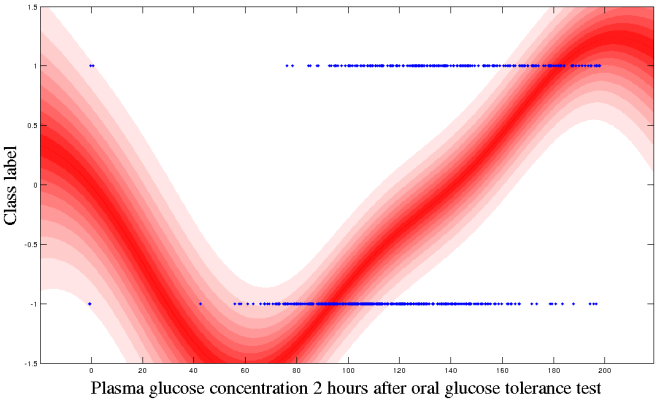


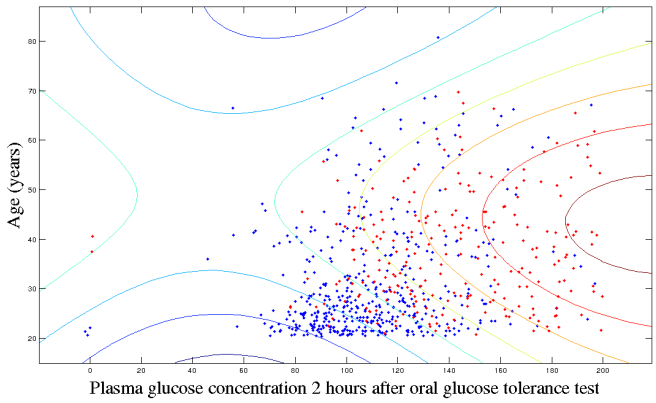


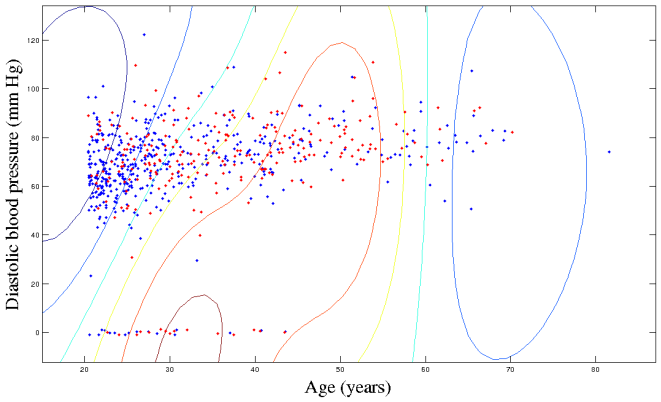


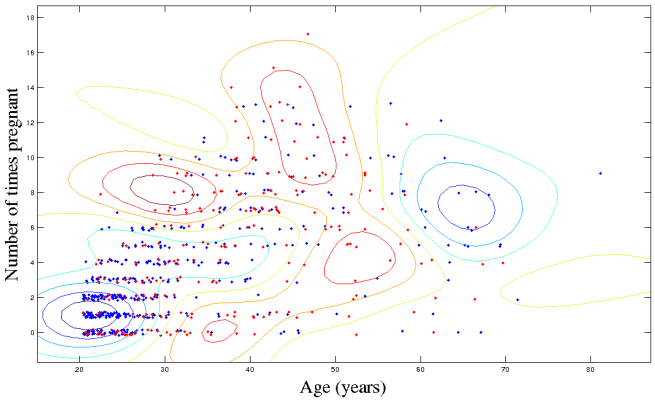


In addition to these additive components, the remaining three folds' structures contained occurances of [2], $[2 \times 3 \times 7]$ and $[2 \times 3 \times 7]$. Showing the three-way dependencies in 2D is not straightforward. Below, we show the posterior plots for [2], $[2 \times 8]$ and $[3 \times 8]$, 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.







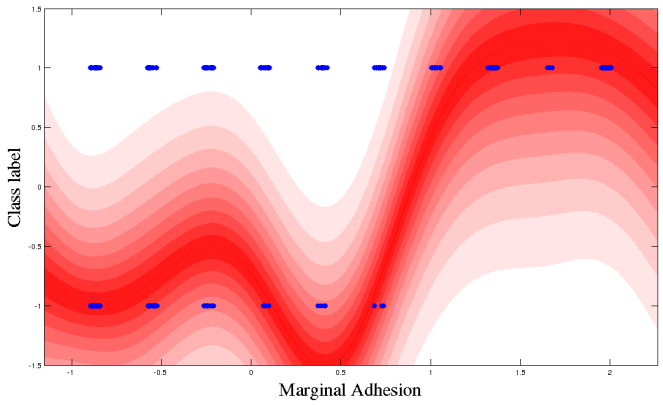


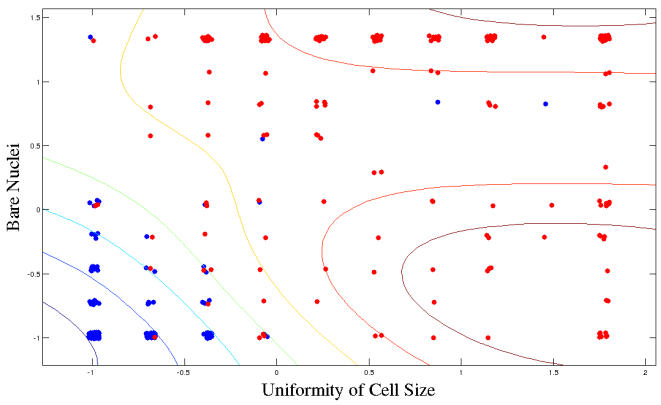
4.7.2 Breast

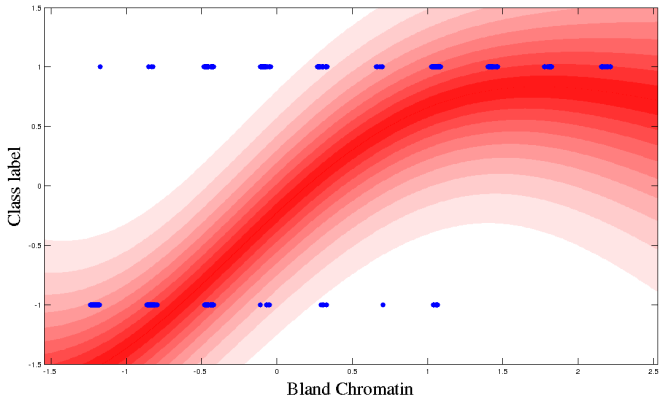
Wisconsin Diagnostic Breast Cancer Dataset is a 9-dimensional dataset which contains measured attributes of 449 individuals, together with an indicator of whether or not they later developed breast cancer. Across the 10 folds generated, there were nine different kernel structures that the procedure returned, 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.

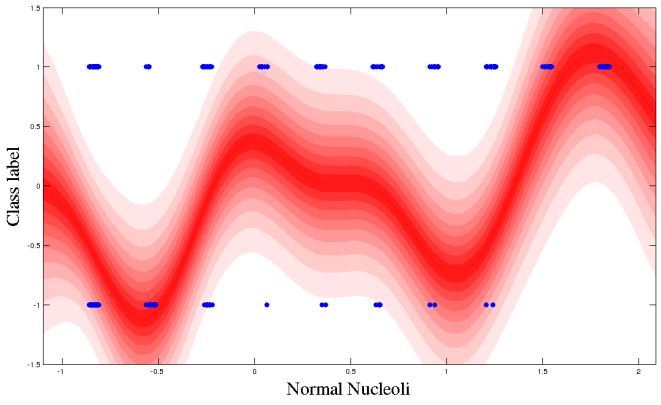
Below, we show those components of the 10 final structures which represent 1 or 2-way interactions. $[1 \times 2 \times 6]$ features in three folds, as do some other high dimensional interactions, but we are still unable to visualise those.

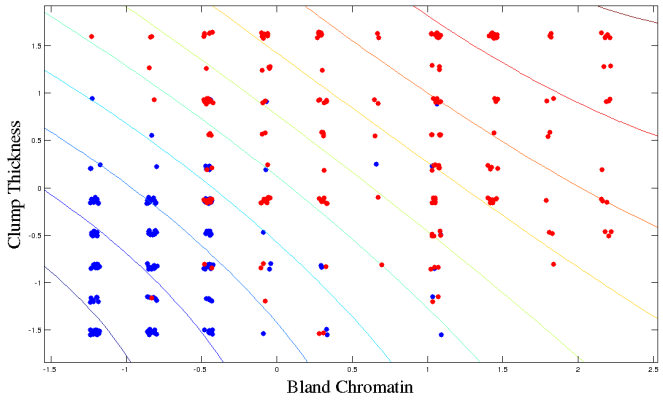
The first six plots show individual components of the final kernel structures obtained using the structure search (not necessarily across the same fold). The final row shows dimensions 1 and 5 using a single SE kernel in the respective dimension.

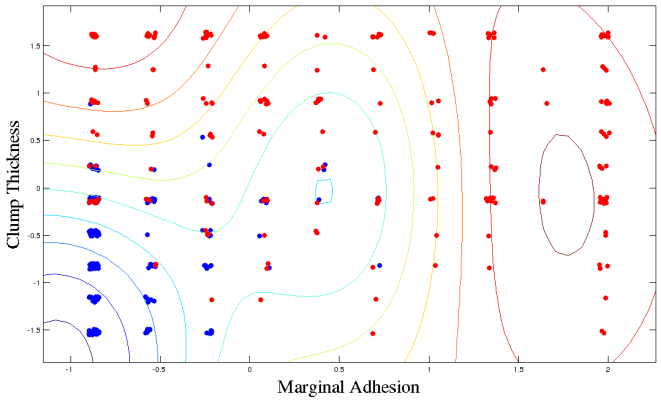

Class label
Marginal Adhesion

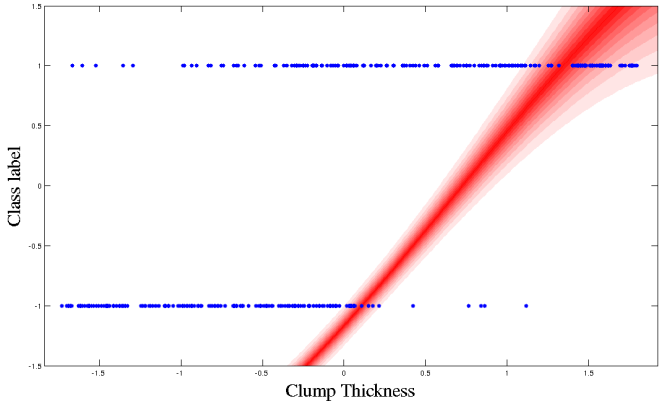

Bare Nuclei
Uniformity of Cell Size

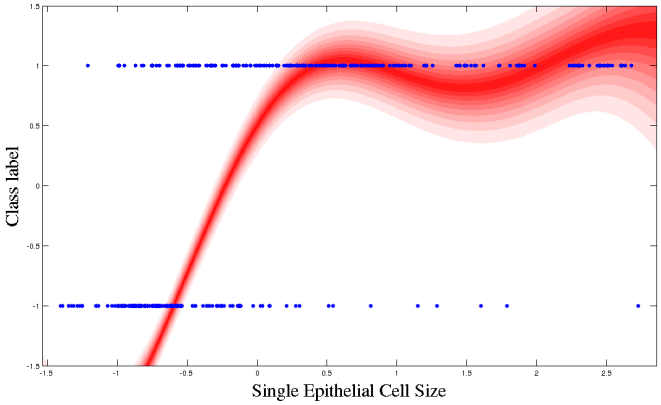

Class label
Bland Chromatin


Class label
Normal Nucleoli


Clump Thickness
Bland Chromatin


Clump Thickness
Marginal Adhesion





4.7.3 Heart and Liver

These two data sets require dealing with representation of discretised data. Plots don't make much sense (yet). Adding jitter might alleviate these issues.

4.8 Automatic Report Synthesis

Chapter 5

Evaluation

In this section, we consider the results achieved by the structure discovery procedure predented in the previous section. We first provide a proof of concept for the procedure by showing that it is able to extract correct kernel structure from synthetic data generated using squared exponential kernels. We then proceed to demonstrate that the performance of the procedure on real world data sets is on par with other state of the art methods such as additive Gaussian Processes and Random Forests. Finally, we present the kernel decompositions on the real world data sets which can be used to uncover and visualise the underlying data patterns which dictate class membership of the data points.

5.1 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 BIC guiding criterion indeed picked the correct model based on marginal likeluihood, the procedure should be able to recover the original kernel used to generate the 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.

True Kernel	N	Kernel recovered (SNR = 1)	Kernel recovered (SNR = 100)
	100	**[1]**	**[1]**
[1]	300	**[1]**	**[1]**
3 dimensions	500	**[1]**	**[1]**
	100	[2]	[2]
[2] + [2] + [2]	300	[2]	[2] + [2]
3 dimensions	500	[2]	[2] + [2]
	100	**[2 × 3]**	**[2 × 3]**
[2 × 3]	300	**[2 × 3]**	**[2 × 3]**
3 dimensions	500	**[2 × 3]**	**[2 × 3]**
	100	[1] + [4]	[1] + [2] + [4]
[1] + [2 × 3] + [4]	300	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
4 dimensions	500	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
	100	[1] + [4]	[1] + [2] + [4]
[1] + [2 × 3] + [4]	300	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
10 dimensions	500	[1 × 4] + [2 × 3]	**[1] + [2 × 3] + [4]**
	100	[4] + [5]	[1 × 9] + [4]
[1] + [2 × 3] + [4] + [5 × 6]	300	[1] + [2] + [4] + [5 × 6 × 7]	[1] + [1 × 5 × 6] + [2 × 3] + [4]
10 dimensions	500	[1] + [1 × 4] + [2 × 3] + [5 × 6]	[1] + [1 × 4 × 5 × 6] + [2 × 3] + [4]
	100	[4 × 7]	**[3 × 5 × 7]**
[3 × 5 × 7]	300	[2 × 3 × 5 × 7]	**[3 × 5 × 7]**
10 dimensions	500	[2 × 3 × 5 × 7]	**[3 × 5 × 7]**
	100	[1 × 3] + [10]	[1 × 10]
[1] + [3 × 5 × 7] + [10]	300	[1] + [10]	[1] + [1 × 10] + [3 × 5 × 7]
10 dimensions	500	[1] + [10]	**[1] + [3 × 5 × 7] + [10]**
	100	[3 × 7]	[1] + [7 × 9]
[3 × 5 × 7 × 9]	300	[7 × 9]	**[3 × 5 × 7 × 9]**
10 dimensions	500	**[3 × 5 × 7 × 9]**	**[3 × 5 × 7 × 9]**
	100	[9 + 10]	[10]
[1] + [3 × 5 × 7 × 9] + [10]	300	[1] + [10]	[1 × 5] + [7] + [10]
10 dimensions	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 are better than the one used to generate the data. The Bayes optimal rates give a theoretical limit on what any model could achieve, as they compare the actual function (y values before adding noise) to the data after the noise is

added. Kernel rate should always be below the Bayes rate...

		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%	

5.1.1 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).

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%	$\mathrm{SE}_1 + \mathrm{SE}_1 \times \mathrm{SE}_3 + \mathrm{SE}_2$	87.0%	91.0%	97.4%
300	100	0%	$\mathrm{SE}_1 + \mathrm{SE}_2 + \mathrm{SE}_3$	94.0%	95.7%	97.4%
500	100	0%	$\mathrm{SE}_1 + \mathrm{SE}_2 + \mathrm{SE}_3$	95.8%	95.4%	97.4%
100	100	5%	$\mathrm{SE}_1 + \mathrm{SE}_2 + \mathrm{SE}_3$	77.0%	80.0%	91.6%
300	100	5%	$\mathrm{SE}_1 \times \mathrm{SE}_3 + \mathrm{SE}_2$	87.0%	85.7%	91.6%
500	100	5%	$\mathrm{SE}_1 \times \mathrm{SE}_2 \times \mathrm{SE}_3$	89.8%	89.8%	91.6%
100	100	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3$	69.0%	69.0%	82.0%
300	100	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3 + \mathrm{SE}_2$	75.3%	73.0%	82.0%
500	100	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3 + \mathrm{SE}_2$	77.6%	74.0%	82.0%
100	1	0%	$\mathrm{SE}_1 + \mathrm{SE}_3$	64.0%	72.0%	77.4%
300	1	0%	$\mathrm{SE}_1 + \mathrm{SE}_3$	74.3%	75.0%	77.4%
500	1	0%	$\mathrm{SE}_1 + \mathrm{SE}_3$	75.6%	76.6%	77.4%
100	1	5%	$\mathrm{SE}_1 + \mathrm{SE}_3$	63.0%	63.0%	74.4%
300	1	5%	$\mathrm{SE}_1 \times \mathrm{SE}_3$	70.7%	68.3%	74.4%
500	1	5%	$\mathrm{SE}_1 \times \mathrm{SE}_3$	72.6%	72.6%	74.4%
100	1	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3$	53.0%	60.0%	68.8%
300	1	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3$	65.3%	65.3%	68.8%
500	1	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3$	66.2%	67.8%	68.8%

Table 5.2: 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%	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_4$	87.0%	92.0%	97.4%
300	100	0%	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_4$	94.0%	94.7%	97.4%
500	100	0%	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_4$	95.6%	96.2%	97.4%
100	100	5%	$\mathrm{SE}_1 + \mathrm{SE}_2$	81.0%	76.0%	92.0%
300	100	5%	$\mathrm{SE}_1 + \mathrm{SE}_2 + \mathrm{SE}_3 \times \mathrm{SE}_4$	85.7%	84.0%	92.0%
500	100	5%	$\mathrm{SE}_1 \times \mathrm{SE}_4 + \mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_3$	87.6%	88.6%	92.0%
100	100	20%	$\mathrm{SE}_2 \times \mathrm{SE}_4$	67.0%	67.0%	82.0%
300	100	20%	$\mathrm{SE}_2 \times \mathrm{SE}_3 + \mathrm{SE}_4$	76.0%	73.7%	82.0%
500	100	20%	$\mathrm{SE}_2 + \mathrm{SE}_3 \times \mathrm{SE}_4$	77.0%	79.8%	82.0%
100	1	0%	SE_2	68.0%	67.0%	76.0%
300	1	0%	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3$	72.3%	70.3%	76.0%
500	1	0%	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3$	72.2%	73.2%	76.0%
100	1	5%	SE_2	67.0%	58.0%	72.2%
300	1	5%	$\mathrm{SE}_1 \times \mathrm{SE}_2$	71.0%	64.3%	72.2%
500	1	5%	$\mathrm{SE}_1 \times \mathrm{SE}_2 \times \mathrm{SE}_3$	70.6%	68.0%	72.2%
100	1	20%	SE_2	59.0%	61.0%	69.0%
300	1	20%	$\mathrm{SE}_2 \times \mathrm{SE}_3 \times \mathrm{SE}_4$	65.3%	62.3%	69.0%
500	1	20%	$\mathrm{SE}_2 \times \mathrm{SE}_3 \times \mathrm{SE}_4$	64.8%	64.8%	69.0%

5.1.2 Saturating Likelihood Function Performance

In the previous chapter, we discussed the use of a mixture of likelihoods that would be better equipped to handle outliers in the data, modelled in this case as the *salt and pepper* noise. In this section, we analyse the performance of the mixture of the error function and the uniform likelihood function on synthetic data.

The use of this likelihood function introduces one new hyperparameter, hyp.lik, which gives us the rates of the two likelihood functions in the mixture. As there are only two likelihood function, we've in fact introduced only one new effective hyperparameter. If the structure search using this likelihood function has potential with helping us deal with outliers, it should be able to estimate the levels of the salt and pepper noise in the synthetic data sets.

TODO: insert tables which represent the performance of the likMix model.

Table 5.3: 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%	$\mathrm{SE}_1 \times \mathrm{SE}_9 + \mathrm{SE}_{10}$	61.0%	88.0%	96.0%
300	100	0%	$\mathrm{SE}_1 + \mathrm{SE}_1 \times \mathrm{SE}_{10} + \mathrm{SE}_3 \times \mathrm{SE}_7$	92.0%	92.7%	96.0%
500	100	0%	$\mathrm{SE}_1 + \mathrm{SE}_1 \times \mathrm{SE}_3 \times \mathrm{SE}_7 \times \mathrm{SE}_{10} + \mathrm{SE}_{10}$	94.2%	94.6%	96.0%
100	100	5%	$\mathrm{SE}_1 \times \mathrm{SE}_9 + \mathrm{SE}_{10}$	53.0%	71.0%	91.8%
300	100	5%	$\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_7 + \mathrm{SE}_6 \times \mathrm{SE}_{10}$	82.0%	81.3%	91.8%
500	100	5%	$\mathrm{SE}_1 \times \mathrm{SE}_3 \times \mathrm{SE}_7 \times \mathrm{SE}_{10} + \mathrm{SE}_{10}$	85.0%	86.2%	91.8%
100	100	20%	SE_1	49.0%	64.0%	79.8%
300	100	20%	$\mathrm{SE}_1 + \mathrm{SE}_{10}$	60.0%	70.0%	79.8%
500	100	20%	$\mathrm{SE}_1 \times \mathrm{SE}_3 \times \mathrm{SE}_7 \times \mathrm{SE}_{10}$	74.2%	75.2%	79.8%
100	1	0%	SE_{10}	59.0%	70.0%	74.4%
300	1	0%	$\mathrm{SE}_1 \times \mathrm{SE}_3 \times \mathrm{SE}_7 \times \mathrm{SE}_{10} + \mathrm{SE}_{10}$	71.3%	72.7%	74.4%
500	1	0%	$\mathrm{SE}_1 \times \mathrm{SE}_{10} + \mathrm{SE}_3 \times \mathrm{SE}_7 + \mathrm{SE}_9$	72.0%	71.4%	74.4%
100	1	5%	SE_{10}	55.0%	66.0%	71.4%
300	1	5%	$\mathrm{SE}_1 \times \mathrm{SE}_{10}$	58.7%	68.7%	71.4%
500	1	5%	$\mathrm{SE}_1 + \mathrm{SE}_{10}$	60.6%	69.4%	71.4%
100	1	20%	SE_3	55.0%	56.0%	65.4%
300	1	20%	SE_{10}	58.0%	61.7%	65.4%
500	1	20%	$\mathrm{SE}_1 \times \mathrm{SE}_{10}$	58.2%	62.0%	65.4%

Table 5.4: 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%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	85.0%	86.0%	97.0%
300	100	0%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7 + \mathrm{SE}_9$	93.7%	93.0%	97.0%
500	100	0%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	91.4%	92.2%	97.0%
100	100	5%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	78.0%	76.0%	91.6%
300	100	5%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	84.0%	83.7%	91.6%
500	100	5%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	86.2%	83.6%	91.6%
100	100	20%	SE_8	49.0%	59.0%	82.0%
300	100	20%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	68.3%	66.0%	82.0%
500	100	20%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	72.2%	66.0%	82.0%
100	1	0%	$\mathrm{SE}_1 \times \mathrm{SE}_3 \times \mathrm{SE}_4 \times \mathrm{SE}_5 + \mathrm{SE}_7$	59.0%	66.0%	74.2%
300	1	0%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7 + \mathrm{SE}_9$	71.7%	72.7%	74.2%
500	1	0%	$\mathrm{SE}_1 + \mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	73.0%	70.6%	74.2%
100	1	5%	$\mathrm{SE}_1 \times \mathrm{SE}_3 \times \mathrm{SE}_4 \times \mathrm{SE}_5 + \mathrm{SE}_7$	55.0%	62.0%	70.8%
300	1	5%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	64.3%	68.7%	70.8%
500	1	5%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_7$	70.4%	67.4%	70.8%
100	1	20%	$\mathrm{SE}_3 \times \mathrm{SE}_5 \times \mathrm{SE}_9$	52.0%	64.0%	66.4%
300	1	20%	$\mathrm{SE}_3 \times \mathrm{SE}_7 \times \mathrm{SE}_8$	55.7%	61.7%	66.4%
500	1	20%	$\mathrm{SE}_3 \times \mathrm{SE}_7$	56.4%	62.6%	66.4%

5.2 Experiments on Real World Data Sets

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 [1] 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 test accuracy. We also show the performance of the kernel using a likelihood mixture to account for outliers.

The table below contains the mean classication error across 10 train-test splits between different methods. The best performing model is shown in bold, together with all other models that were not significantly different from it, according to the paired t-test for statistical significance. 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.

Table 5.5: 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 (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**

Chapter 6

Summary and Conclusions

6.1 Further Work

Bibliography

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