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 2, 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 Bayesian Machine Learning in the context of Data Science

1.2 Contribution

Chapter 2

Gaussian Processes

2.1 The Role of 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, with different models.

3.2 Kernel structure discovery for regression

Unsupervised structure discovery - mention that paper, maybe initially, as a similar type of work... 1 paragraph?

Talk about the fact that these methods fix the structure of the kernel beforehand. Do they? Or they lack interpretability.

As a critique, 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.

Write an intro to doing this for classification, why it is harder (less information than regression, harder interpretability, non-Gaussian likelihoods and the need to resort to nlml approximations).

3.3 Additive Gaussian Processes

Find other relevant work - Dave mentioned some of these. Read through all the papers' relevant work sections.

Chapter 4

Kernel Structure Discovery for GP Classification

4.1 Defining the Kernel Grammar

Draw the basic kernels, describe additive and product kernels.

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.

4.2 Model selection

4.3 Optimising the Hyperparameters

Subsampling the training data to optimize the process.

Discuss parallelisation. Maybe add a table of running times and numbers of restarts.

4.3.1 Overfitting

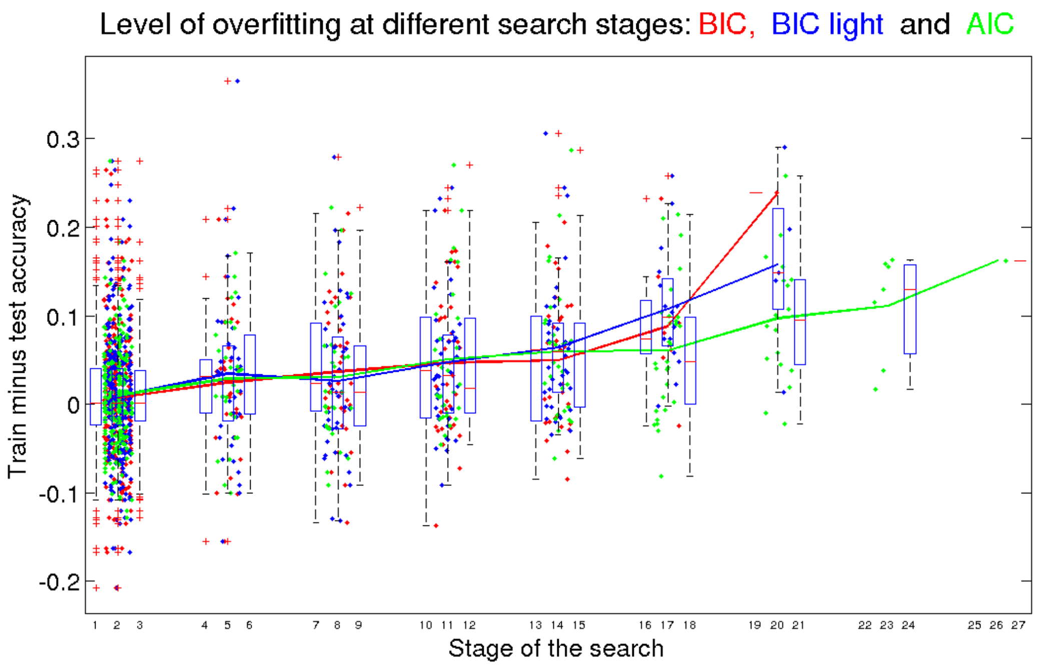
Dealing with very small and very large hyperparameters.

4.4 Guiding the Structure Search

4.4.1 Bayesian and Akaike Information Criteria

4.4.2 The Number of Effective Hyperparameters

Outline BIC light, present it as middle ground between full BIC and AIC. Insert the number of search steps figure that shows overfitting.



4.4.3 Cross-validated training accuracy

4.5 Adapting the likelihood function

4.5.1 Dealing with Outliers

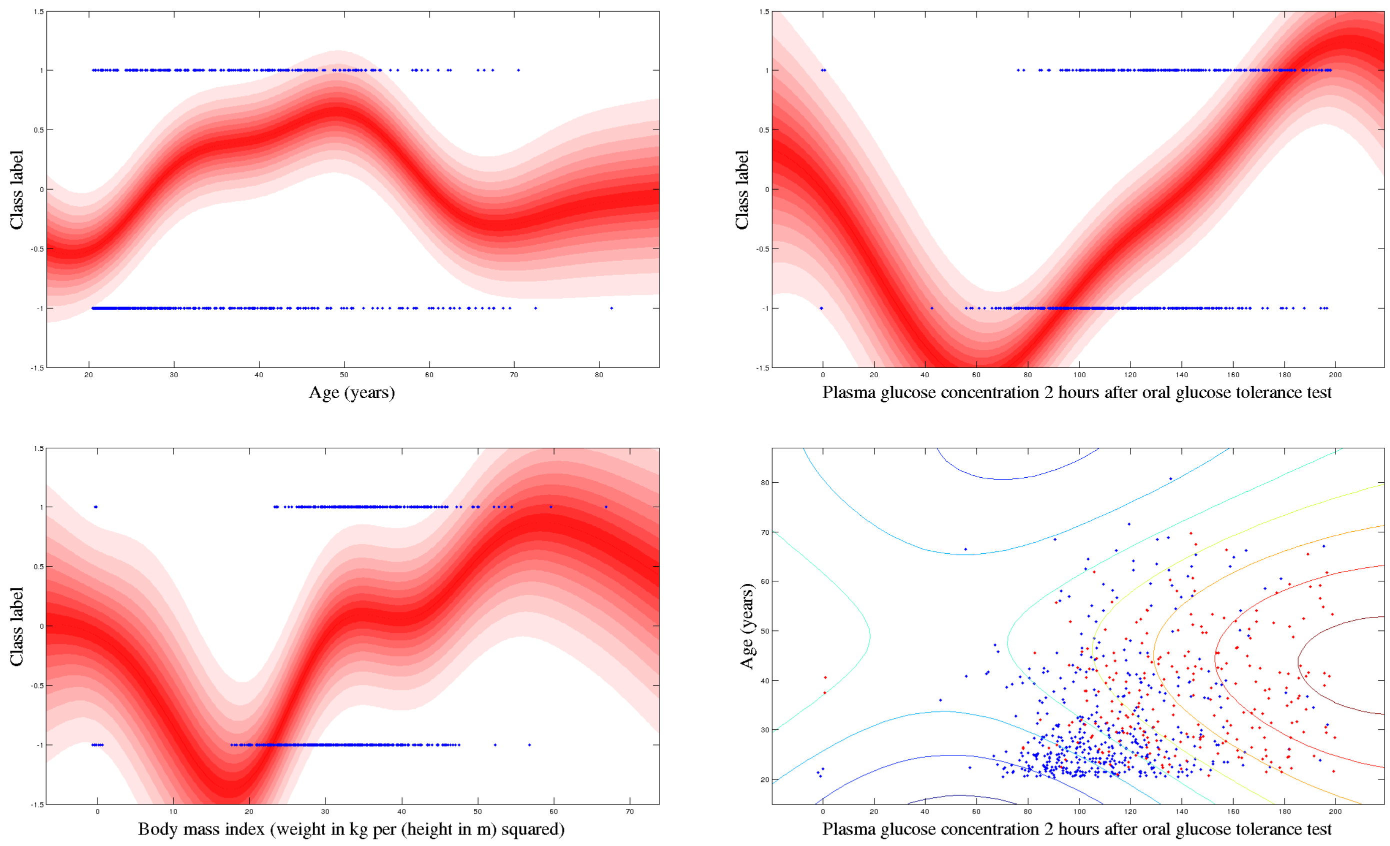
a problem with classification and BIC is that estimating the number of effective hyperparameters is hard: not all of them affect the classification boundary equally. The only thing important for classification is the zero-crossings of the function (where it is greater, and where it is smaller than 0, 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).

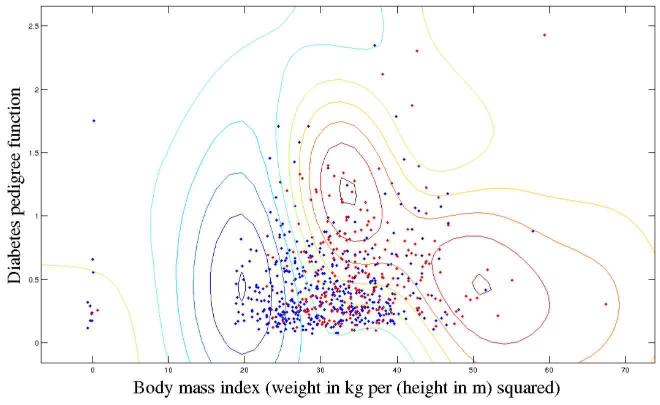
The reason that GPs out of the box are easily outperformed by classification specific schemes such as SVMs is that soft-margin SVMs are less certain of their predictions, and hence are less certain about points far away from the high density regions of the data. SVMs are never more certain than the estimated 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...

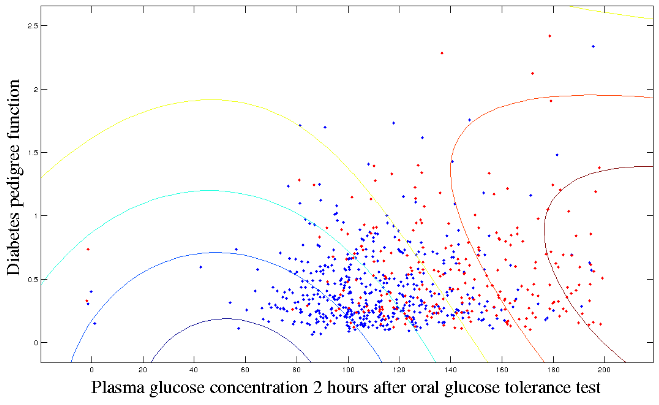
As a potential solution, we might want to use a @likMix which adds the pepper noise into the likelihood function, determining the level of the outliers and thus having the likelihood (i.e. sigmoid) which doesnt range from 0 to 1, but from α to $(1-\alpha)$, where α is the level of pepper noise, which 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.

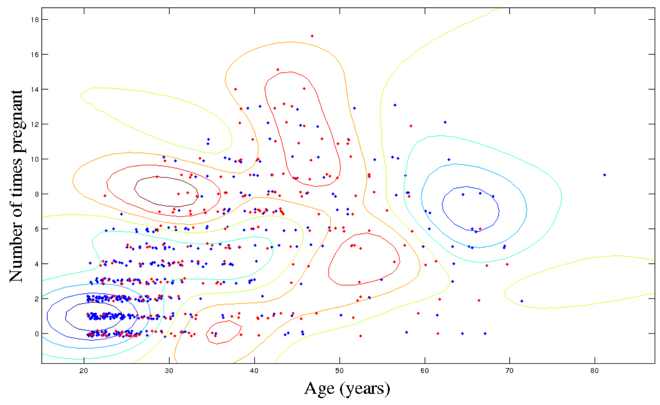
4.6 Providing Interpretability

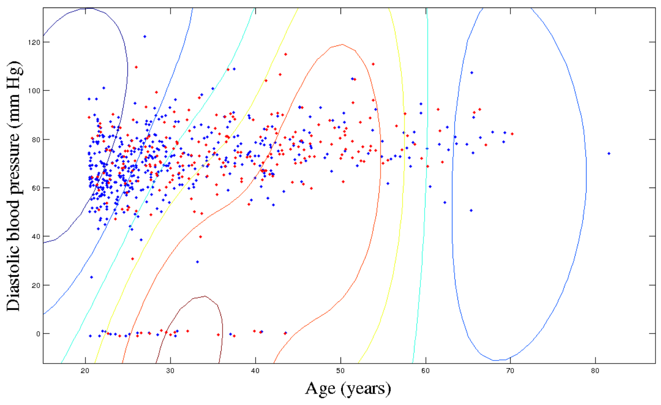
4.6.1 Pima











4.6.2 Liver

4.6.3 Heart

4.6.4 Visualising the kernel decomposition

4.7 Bayesian Model Averaging

4.7.1 BMA for Predictive Performance

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

Discuss assigning different alphas, make a plot showing how the accuracy changes.

4.7.2 BMA for Model Selection

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]
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	[4]	**[1] + [2 × 3] + [4]**
[1] + [2 × 3] + [4]	300	[2 × 3] + [4]	**[1] + [2 × 3] + [4]**
4 dimensions	500	**[1] + [2 × 3] + [4]**	**[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] + [5 × 6] + [4]	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	**[3 × 5 × 7]**	**[3 × 5 × 7]**
[3 × 5 × 7]	300	**[3 × 5 × 7]**	**[3 × 5 × 7]**
10 dimensions	500	**[3 × 5 × 7]**	**[3 × 5 × 7]**
	100	[1 × 3] + [10]	[1 × 10]
[1] + [3 × 5 × 7] + [10]	300	[1] + [10]	[1 × 10] + [3 × 5 × 7] + [10]
10 dimensions	500	[1] + [10]	**[1] + [3 × 5 × 7] + [10]**
	100	[1]	[1] + [7 × 9]
[3 × 5 × 7 × 9]	300	[3 × 5] + [7 × 9]	**[3 × 5 × 7 × 9]**
10 dimensions	500	**[3 × 5 × 7 × 9]**	**[3 × 5 × 7 × 9]**
	100	[10]	[1] + [3 × 5 × 7]
[1] + [3 × 5 × 7 × 9] + [10]	300	[1] + [10]	[3 × 5 × 7 × 9]
10 dimensions	500	[1] + [3 × 5 × 9] + [10]	**[1] + [3 × 5 × 7 × 9] + [10]**

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: $\text{SE}_1 + \text{SE}_2 + \text{SE}_3$, $D = 3$.

Data size	SNR	sp_noise	Kernel chosen	Test accuracy	Kernel rate	Bayes rate
100	100	0%	$\text{SE}_1 + \text{SE}_1 \times \text{SE}_3 + \text{SE}_2$	87.0%	91.0%	97.4%
300	100	0%	$\text{SE}_1 + \text{SE}_2 + \text{SE}_3$	94.0%	95.7%	97.4%
500	100	0%	$\text{SE}_1 + \text{SE}_2 + \text{SE}_3$	95.8%	95.4%	97.4%
100	100	5%	$\text{SE}_1 + \text{SE}_2 + \text{SE}_3$	77.0%	80.0%	91.6%
300	100	5%	$\text{SE}_1 \times \text{SE}_3 + \text{SE}_2$	87.0%	85.7%	91.6%
500	100	5%	$\text{SE}_1 \times \text{SE}_2 \times \text{SE}_3$	89.8%	89.8%	91.6%
100	100	20%	$\text{SE}_1 \times \text{SE}_3$	69.0%	69.0%	82.0%
300	100	20%	$\text{SE}_1 \times \text{SE}_3 + \text{SE}_2$	75.3%	73.0%	82.0%
500	100	20%	$\text{SE}_1 \times \text{SE}_3 + \text{SE}_2$	77.6%	74.0%	82.0%
100	1	0%	$\text{SE}_1 + \text{SE}_3$	64.0%	72.0%	77.4%
300	1	0%	$\text{SE}_1 + \text{SE}_3$	74.3%	75.0%	77.4%
500	1	0%	$\text{SE}_1 + \text{SE}_3$	75.6%	76.6%	77.4%
100	1	5%	$\text{SE}_1 + \text{SE}_3$	63.0%	63.0%	74.4%
300	1	5%	$\text{SE}_1 \times \text{SE}_3$	70.7%	68.3%	74.4%
500	1	5%	$\text{SE}_1 \times \text{SE}_3$	72.6%	72.6%	74.4%
100	1	20%	$\text{SE}_1 \times \text{SE}_3$	53.0%	60.0%	68.8%
300	1	20%	$\text{SE}_1 \times \text{SE}_3$	65.3%	65.3%	68.8%
500	1	20%	$\text{SE}_1 \times \text{SE}_3$	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%

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**

5.3 Visualisation

Describe what is being plotted and at what stage.

Pima example:

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.