

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

<http://wrap.warwick.ac.uk/137225>

Copyright and reuse:

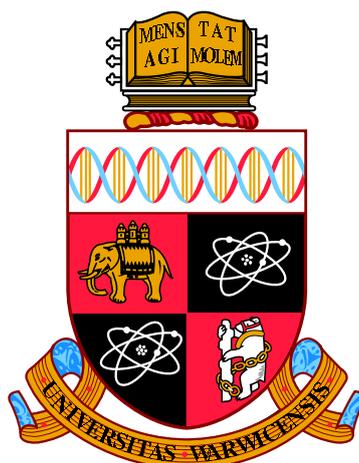
This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it.

Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk



**Machine Learning Techniques for the Early
Detection of Cancer using Volatile Organic
Compounds**

by

Matthew David Neal

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Statistics

January 2019

THE UNIVERSITY OF
WARWICK

Contents

List of Tables	v
List of Figures	vii
Abbreviations	x
Acknowledgements	xiii
Declarations	xiv
Abstract	xv
Chapter 1 Introduction	1
1.1 Overview	1
1.2 Motivation: early cancer detection	3
1.3 Aims and scope	4
1.4 Methodology	5
1.4.1 Assessing classifier performance	5
1.4.2 Metrics	7
1.4.3 Classification models	9
1.4.4 Gaussian processes	10
1.4.5 Probabilistic PCA	11
1.4.6 The Gaussian process latent variable model (GPLVM)	12
1.4.7 Stochastic gradient descent	12
1.5 Overview	13
Chapter 2 Volatile Organic Compounds: A Novel Biomarker for Low-cost Cancer Detection	15
2.1 Overview	15
2.1.1 Volatile Organic Compounds (VOCs)	15

2.1.2	The “Two Week Wait” urgent care pathway	18
2.1.3	Other studies on detecting colorectal cancer with urinary VOCs	18
2.2	Challenges of working with FAIMS data	19
2.2.1	Failed samples and outlier detection	20
2.2.2	Assigning autosampled data to sample IDs	23
2.2.3	High-dimensional data	23
2.2.4	Study design considerations	25
2.3	Methods	27
2.3.1	Managing FAIMS data	27
2.3.2	Importing FAIMS data into R	30
2.3.3	Outlier detection, denoising & data transformation	30
2.3.4	Visualisations of FAIMS data	30
2.3.5	Classifier training and validation	30
2.3.6	Classification models	34
2.4	Experimental results	36
2.4.1	UHCW Two Week Wait colorectal cancer pathway	37
2.4.2	Colorectal cancer screening using urinary VOCs	49
2.4.3	Distinguishing colorectal cancer from IBS	53
2.4.4	Pre-symptomatic detection of anastomotic leakage following gastrointestinal surgery	55
2.4.5	Detection of anastomotic leakage following major colorectal surgery	61

Chapter 3 Gaussian Process Latent Variable Models - Theory and Literature Review **69**

3.1	Overview	69
3.2	Background Theory	70
3.2.1	Gaussian processes	70
3.2.2	Reproducing kernel Hilbert spaces - a brief foray into functional analysis	74
3.2.3	Kernel functions	75
3.2.4	Hyperparameter optimization	81
3.2.5	PCA, probabilistic PCA, and dual probabilistic PCA	85
3.2.6	Gaussian process latent variable models	87
3.3	Recent developments in Gaussian processes	88

Chapter 4	Structured Gaussian Process Latent Variable Models	91
4.1	Structured GPLVM	91
4.1.1	Formulation of structured GPLVM	91
4.1.2	Examples of structure spaces	93
4.2	Stochastic optimization	95
4.2.1	Intractability and wild geese	95
4.2.2	Stochastic gradient descent	96
4.2.3	Problems with optimizing SGPLVM using SGD	97
4.2.4	Improvements to SGD	98
4.2.5	Introducing back-constraints	101
4.3	Experiments on artificial data	102
4.4	Stochastic gradient descent variants	103
4.5	Comparison to GPLVM	110
4.5.1	Comparison of exact SGPLVM to a published GPLVM imple- mentation	110
4.5.2	Comparison of approximate SGPLVM to GPLVM	111
4.6	Impact of approximation size	113
4.7	Partially observed data	116
4.8	Experiments on FAIMS data	117
Chapter 5	SGPLVM and GP Regression: Implementation & Exten- sions	125
5.1	The gaussianProcess package	125
5.1.1	Kernel Model Trees	125
5.1.2	Learning the optimum kernel	126
5.1.3	Assessing model fit	130
5.1.4	Kernels implemented	132
5.1.5	Comparison to other R packages for fitting Gaussian processes	133
5.1.6	Caching results in R to improve performance	133
5.2	The GPLVM package	134
5.2.1	Approximating a Hessian vector product	134
5.2.2	Probabilistic estimation of the trace of a matrix product . . .	137
5.2.3	Piecewise exact calculation of the trace of a matrix product .	138
5.2.4	Priors on latent space coördinates	139
5.2.5	Representing structured data in R	139
5.2.6	Efficiently sampling from an SGPLVM prior	140
5.2.7	Discriminative priors for SGPLVM	140

5.3 Summary	141
Chapter 6 Conclusion	145
Appendix A TWW Study JAGS Models	149
A.1 Logistic regression	149
A.2 Logistic regression with interactions	150
A.3 Robust logistic regression	151
A.4 Independent robust mixture model	152
A.5 Independent robust mixture model with implicit subclasses and censoring	153
Appendix B FAIMS Autosampler Data Auto-splitter	157
B.1 readme.txt	157
Appendix C FAIMSToolkit documentation	159
Appendix D Mathematical Background	185
D.1 Matrix identities and derivatives	185
D.1.1 The Woodbury matrix identity	185
D.1.2 Proof	185
D.1.3 Matrix derivatives	186
D.2 Reproducing Kernel Hilbert Spaces	186
D.2.1 Definitions	186
D.2.2 Equivalence of definitions	187
D.2.3 The Moore–Aronszajn theorem	189

List of Tables

2.1	Table of NPV, sensitivity, and specificity for the different models for combining F-Hb and FC results, as well as for raw F-Hb results.	42
2.2	Table of NPV, sensitivity, and specificity for the different models for combining F-Hb and FC results with additional clinical information, as well as for raw F-Hb results without any clinical information for comparison.	42
2.3	Compliance-adjusted statistics for F-Hb and urinary VOCs for detecting colorectal cancer in the TWW study.	48
2.4	AUC for a number of classifiers for the detection of colorectal disease using urinary VOCs.	50
2.5	Power calculations for colorectal cancer screening study	53
2.6	Model selection results for the anastomotic leakage pilot study.	58
2.7	Results for training a Random Forest classifier on urinary VOC data to detect anastomotic leakage after surgery.	59
2.8	Results for training a Random Forest classifier on urinary VOC data to detect anastomotic leakage after major colorectal surgery.	65
2.9	Results for the standard analysis pipeline for detection of AL.	67
4.1	Generating parameters for artificial datasets.	103
4.2	Convergence performance of various SGD algorithms.	109
4.3	Comparison of performance of PCA, GPLVM, SGPLVM, and SG-PLVM with automatic dimensionality selection.	112
4.4	RMSE of the reconstructed latent space for a range of approximation sizes (ARD).	116
4.5	RMSE of the reconstructed latent space for a range of approximation sizes (2D)	117
4.6	RMSE for recovery of the latent space in the presence of missing data.	120
4.7	ROC AUCs for 2D latent spaces of FAIMS studies.	124

5.1	BIC and RMSE for compound kernels fitted to the Mauna Loa atmospheric CO ₂ concentration data	129
-----	--	-----

List of Figures

2.1	Example visualisation of FAIMS data.	17
2.2	Examples of FAIMS samples which have low flow rates reported in their associated metadata.	21
2.3	A plot of the mean kNN distance for the data in the anastomotic leakage pilot study.	22
2.4	Example output from the tool we developed to automatically identify in-sample runs and assign sample IDs to autosampled FAIMS data.	24
2.5	Example results for identifying 60% of variables as noise using standard deviation and mean correlation to neighbouring pixels.	26
2.6	Activity diagram showing the typical use of our FAIMS software.	28
2.7	Activity diagram showing the typical data flow when using our FAIMS software.	29
2.8	Example output from <code>plotFAIMSdata</code> for six samples.	31
2.9	Plot of the number of principal components retained against the evidence, and a scree plot of the eigenvalue associated with each principal component.	32
2.10	ROC curves for the prediction of colorectal cancer (CRC), high risk adenomas (>10mm in diameter), and all adenomas using thresholding of F-Hb. 95% confidence intervals are shown in grey.	40
2.11	ROC curves for the prediction of colorectal cancer (CRC), high risk adenomas (>10mm in diameter), and all adenomas using thresholding of FC. 95% confidence intervals are shown in grey.	40
2.12	ROC curves for the six models investigated for combining F-Hb and FC to predict colorectal cancer.	41
2.13	Predicted probability of cancer for the different methods of combining F-Hb and FC.	43
2.14	ROC curves for the three models investigated for combining F-Hb, FC and clinical data to predict colorectal cancer.	43

2.15	ROC curves for the prediction of colorectal cancer (CRC), high risk adenomas (>10mm in diameter), and all adenomas using VOCs. . .	44
2.16	Example bootstrap distributions for the NPV, sensitivity, and specificity of F-Hb testing, adjusted to take into account the compliance rate of the test.	47
2.17	Predicted probability of polyps by true disease.	51
2.18	Power calculations for colorectal cancer screening study	52
2.19	Plots for detecting FAIMS drift.	54
2.20	Example urinary VOC data for patients with and without AL following pancreaticoduodenectomy.	57
2.21	ROC curves for detecting anastomotic leakage after surgery using urinary VOCs and Random Forests.	59
2.22	Predicted probability of anastomotic leakage broken down by surgery type.	60
2.23	Predicted probability of anastomotic leakage for pancreaticoduodenectomy patients, broken down by the number of days after surgery that the urine sample was provided.	60
2.24	Variables excluded by standard deviation thresholding in the AL study.	63
2.25	ROC curves for detection of anastomotic leakage after major colorectal surgery.	64
2.26	Variable importance for Random Forest classification of anastomotic leakage after major colorectal surgery.	66
3.1	Examples of functions drawn from GP priors with linear and polynomial kernels.	75
3.2	Examples of the effect of varying the hyperparameters of the squared exponential and periodic kernels.	77
3.3	Examples of functions sampled from a GP prior with a neural network kernel.	78
3.4	Contour plot of GP log marginal likelihood as a function of the kernel hyperparameters.	83
4.1	Artificial data used for assessing SGPLVM performance.	104
4.2	Artificial data for assessing SGD algorithms	105
4.3	Likelihood convergence for standard SGD.	106
4.4	Likelihood convergence for SGD with momentum.	106
4.5	Likelihood convergence for SMD.	107
4.6	Likelihood convergence for Adam.	107

4.7	Comparison of likelihood convergence for different SGD algorithms. .	108
4.8	Comparison of SGPLVM to a published implementation of GPLVM.	110
4.9	Reconstructed latent space RMSE for GPLVM versus SGPLVM for twenty datasets	111
4.10	Latent space recovery on the distinct-clusters data using PCA and SGPLVM.	113
4.11	Improvement over PCA of the recovered latent space RMSE for a range of approximation sizes over the twenty datasets generated from overlapping clusters using ARD SGPLVM.	114
4.12	Improvement over PCA of the recovered latent space RMSE for a range of approximation sizes over the twenty datasets generated from overlapping clusters using 2D SGPLVM.	115
4.13	Examples of partially observed artificial data.	118
4.14	The recovered latent space for partially observed data with distinct clusters.	118
4.15	Mean SGPLVM-recovered latent space RMSE for twenty datasets generated from a single true latent space with overlapping clusters, plotted against the proportion of observed data missing.	119
4.16	SGPLVM hyperparameter convergence for the two AL studies	122
4.17	Recovered latent spaces for AL Studies 1 & 2.	123
5.1	Modelling compound kernels using model trees.	127
5.2	Selected compound kernels for the Mauna Loa atmospheric CO ₂ concentration data.	128
5.3	Effect of discriminative priors on the learned latent representation in SGPLVM.	142

Abbreviations

AL anastomotic leakage. 55, 56, 57, 59, 61, 62, 64

ARD automatic relevance determination. 76, 92, 132

AUC area under the curve. 9, 42, 44, 48, 50, 63, 64, 145

BCa bias corrected and accelerated. 50

BFGS Broyden–Fletcher–Goldfarb–Shanno. 12, 84

BIC Bayesian information criterion. 131

CSV comma-separated values. 103

ctDNA circulating tumour DNA. 3

ECCG Esophagectomy Complications Consensus Group. 56

FC faecal calprotectin. 37, 38, 39, 41, 42, 43, 44, 48

F-Hb faecal haemoglobin. 37, 38, 39, 41, 42, 43, 44, 46, 47, 48, 49

FAIMS field asymmetric ion mobility spectrometry. 2, 4, 5, 9, 13, 16, 18, 37, 49,
53, 55, 56, 61, 62, 91, 117, 139, 145, 146, 147, 148

FIT faecal immunochemical tests. 37, 44, 145, 148

FOB faecal occult blood. 4, 37, 45

GC-qMS gas chromatograph quadrupole mass spectrometer. 19

GP Gaussian process. 10, 11, 12, 70, 73, 74, 75, 78, 81, 82, 83, 87, 88, 91, 93, 95,
96, 133, 147, 192

GPC Gaussian process classifier. 147, 148

GPLVM Gaussian process latent variable model. 12, 13, 69, 84, 87, 91, 92, 93, 101, 102, 125, 139, 140, 143, 145, 146, 147

GUI graphical user interface. 34

IBS irritable bowel syndrome. 19, 37, 53

IMS ion mobility spectrometry. 16

IQR interquartile range. 57, 58

IRMM independent robust mixture models. 38

IRMM-IS-C independent robust mixture models with implicit subclasses and censoring. 38, 39, 41

ISGPF International Study Group on Pancreatic Fistula. 56

IUPAC International Union of Pure and Applied Chemistry. 15

IVM informative vector machine. 134

k-NN k nearest neighbours. 20, 21, 22, 30, 33, 81

LDA linear discriminant analysis. 2, 7, 19

LOOCV leave-one-out cross-validation. 6

MAP maximum a posteriori. 130, 131

MS mass spectrometry. 16

MWW Mann-Whitney-Wilcoxon. 9, 31, 39, 50, 64

NHS National Health Service. 4

NICE National Institute for Health and Care Excellence. 18, 37, 38

NPV negative predictive value. 8, 38, 42, 44, 45, 47, 48

PCA principal component analysis. 2, 11, 32, 39, 85, 87

PPCA probabilistic PCA. 11, 12, 85, 86

PPV positive predictive value. 8, 18, 45

RKHS reproducing kernel Hilbert space. 71, 74, 75, 187, 189

RMSE root-mean-square error. 102

ROC receiver operating characteristic. 9, 42, 63

SGD stochastic gradient descent. 12, 13, 96, 97, 98, 99, 103, 146

SGoF sequential goodness of fit. 31

SGPLVM structured Gaussian process latent variable model. 2, 3, 13, 91, 96, 97, 98, 100, 101, 125, 139, 140, 143, 145, 146, 147

SMD stochastic meta descent. 99, 103

SVD singular value decomposition. 85, 87

SVM support vector machine. 10, 33, 35, 36, 57, 58, 70, 81

TWW Two Week Wait. 18, 37, 38, 49

UHCW University Hospitals Coventry & Warwickshire. 18, 37, 38, 44, 48, 49

VOC volatile organic compound. 1, 2, 4, 13, 15, 16, 18, 19, 36, 37, 42, 44, 46, 48, 49, 50, 55, 61, 62, 64, 145, 148

VUmc VU University Medical Center. 37, 56, 62

Acknowledgements

First and foremost, thank you to my supervisor Rich Savage, without whom this thesis would not have been possible. His support, knowledge, and kind words have been invaluable.

I am grateful to James Covington for his advice and insight over the past years. Thanks also to our collaborators, Ramesh Arasaradnam, Monika Widlak, Ella Mozdiak, Nidhi Sagar, and Victor Plat; it has been a pleasure and a privilege working with you.

Thank you to Sascha Ott, for teaching me to teach; thanks also to the friends I made in my itinerant journey through the departments of the University of Warwick, for many wonderful conversations.

Finally, thank you to my wife, Emma Blamey, for her unwavering support, and to my son Eliot, for making me smile when nothing else did.

This work was supported by the Engineering and Physical Sciences Research Council as part of the University of Warwick Systems Biology Doctoral Training Centre.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.

The work presented (including data generated and data analysis) was carried out by the author except in the cases outlined below:

- The data from the study discussed in section 2.4.1 was kindly provided by our University Hospitals Coventry and Warwickshire (UHCW) collaborators Dr Ramesh Arasaradnam and Dr Monika Widlak.
- The data from the study discussed in section 2.4.2 was kindly provided by our UHCW collaborators Dr Ramesh Arasaradnam and Dr Ella Mozdiak.
- The data discussed in section 2.4.3 was kindly provided by our UHCW collaborators Dr Ramesh Arasaradnam, Dr Ella Mozdiak, and Dr Nidhi Sagar.
- The data discussed in section 2.4.4 and 2.4.5 and revisited in section 4.8 was kindly provided by our collaborator Dr Victor Plat of VU University Medical Center.

Parts of this thesis have been published by the author:

- The work in section 2.4.1 has been published as part of Widlak et al. [2018].
- The work in section 2.4.4 has been published as part of Plat et al. [2018].
- The work in section 2.4.5 has been included in a paper which has been submitted for publication.

Abstract

Early cancer detection can change lives. If cancer can be detected early in its development, while a patient is still asymptomatic, it is easier and less expensive to treat. Volatile organic compounds, as measured by field asymmetric ion mobility spectrometry (FAIMS), are a novel class of biomarkers which have shown promise as a low-cost early screening test for a range of cancers. However, FAIMS data is high-dimensional, difficult to interpret, and can be subject to a range of subtle data quality problems. Additionally, in the current literature many researchers use linear methods for analysing FAIMS data. We believe improved results could be achieved by applying modern machine learning techniques to the problem.

In this thesis, we investigate using modern machine learning techniques and best practices for FAIMS analysis, and develop corresponding software. We found that FAIMS has a moderate ability to detect cancer in an at-risk population, and that FAIMS could be used for the pre-symptomatic detection of other diseases with excellent results, achieving an AUC of 0.91 for the pre-symptomatic detection of anastomotic leakage after surgical resection to treat cancer.

We present a novel Bayesian dimensionality reduction technique, the structured Gaussian process latent variable model (SGPLVM), which extends GPLVM to exploit structured correlations between variables, as are seen in FAIMS data. We also present a stochastic optimization algorithm and a number of extensions based on stochastic gradient descent variants. We explore the properties of SGPLVM, which we found to outperform GPLVM at recovering a latent representation of data which meet the model assumptions. We also demonstrate SGPLVM's robustness to partially observed data.

Finally, we present software packages for GPLVM, SGPLVM, and GP regression with a novel method of specifying and automatically selecting compound kernel functions.

Chapter 1

Introduction

1.1 Overview

The early detection of cancer is an extraordinarily fertile area of research, and with good reason: early detection of cancer leads to improved outcomes for patients and reduces the cost of treatment [Smith et al., 2010]. A recent systematic review of 19,724 papers published between 2010 and 2014 found 3990 papers on the subject of biomarkers for the early detection of cancer which met the inclusion criteria,¹ covering 814 potential blood-based biomarkers [Uttley et al., 2016]. Unfortunately, few biomarkers make the transition into clinical practice [Cree, 2015; Day, 2016]. To be used in clinical practice, a biomarker must first be shown to provide improved clinical outcomes in a randomized clinical trial. However, most biomarkers do not make it to this stage, with many which show promise initially ultimately not being reproducible [Ransohoff, 2008].

Volatile organic compounds (VOCs) are a novel class of biomarkers which are measurable in a wide range of sample mediums [O'Hara et al., 2009; Hanai et al., 2012; Soini et al., 2010; Dixon et al., 2007; Phillips et al., 1999], and have shown promise in the early detection of cancer [Wehinger et al., 2007; Bajtarevic et al., 2009; Westhoff et al., 2009; Arasaradnam et al., 2014; Machado et al., 2005; Peng et al., 2010; Westenbrink et al., 2015; Mazzone et al., 2007; Cornu et al., 2011; Buszewski et al., 2012]. They are generated as a by-product of cell metabolism [Amann et al., 2014], and can therefore be considered a metabolomic approach to cancer detection. VOCs have seen widespread use in other fields [Kolakowski and Mester, 2007], and a range of well established instruments exist for measuring VOCs. Since VOCs are

¹Controlled studies of blood-based biomarkers for the detection or diagnosis of cancer.

readily carried in the blood, and tumours with metastatic potential generally depend on having a good blood supply to support their growth [Nishida et al., 2006], VOCs offer promise as a method for liquid biopsy of tumour metabolomes.

While much research has focussed on identification of specific VOCs which are associated with cancer [Di Lena et al., 2016; Silva et al., 2011], these rely on analytic techniques which are too expensive to be used in a screening test. Field asymmetric ion mobility spectrometry (FAIMS) [Kolakowski and Mester, 2007] offers a low-cost method to measure the entire VOC profile of a sample, but does so in a way which is not easily mapped onto the specific compounds which are present. The signals from different compounds in FAIMS data generally overlap, creating a complex and noisy signal. It seems plausible that applying modern machine learning tools to FAIMS data will allow us to access the significant amount of information in the full profile of VOCs. Previous research on urinary FAIMS analysis for the early detection of cancer has not tended to make use of modern developments in machine learning, generally using linear modelling techniques such as principal component analysis (PCA) and linear discriminant analysis (LDA). They have also tended to fall foul of subtle mistakes in the design of their statistical analyses, which introduces bias into the results and makes the true utility of VOCs for cancer detection hard to ascertain.

We developed improved techniques for working with VOC data derived from FAIMS analysis. First, we developed a flexible end-to-end analysis pipeline for classification problems involving FAIMS data, using established machine learning techniques and best practice. We then validated these techniques in a number of studies investigating the use of FAIMS data in a clinical setting, with an emphasis on cancer detection. We also implemented the methods in a publicly available R [R Core Team, 2018a] package, with the intention that this package would allow researchers to analyse FAIMS data in a reproducible way using methods which follow best practice for avoiding bias in the reported results.

Our work on FAIMS data led to the development of a novel Bayesian model for data with structured correlations between variables (of which FAIMS data is an example),² which we call the structured Gaussian process latent variable model (SGPLVM). Alongside this, we present a method for fitting the model to data using stochastic optimization. SGPLVM is a non-linear dimensionality reduction technique which

²An approachable, everyday example of data with structured correlations between observed variables is image data, where there is a prior expectation that any two pixels in close proximity in the image will have correlated values.

incorporates a prior encoding for structured correlations between observed variables. We ran computational experiments against SGPLVM to explore its properties.

1.2 Motivation: early cancer detection

There are a number of examples of successful screening programmes designed to increase the early detection of cancer which have been shown to improve outcomes and reduce the total cost of treatment [Smith et al., 2010]. For example, it has been estimated that the UK’s cervical screening programme prevents more than 5,000 deaths per year from cervical cancer [Peto et al., 2004]. However, many current procedures for early detection of cancer can be expensive to perform, and some early cancer screening programmes suffer from poor uptake [Smith et al., 2010; McGregor et al., 2007]. Many common cancers simply have no viable tests for early detection [Cohen et al., 2018].

A number of studies have investigated possible liquid biopsies for cancer diagnosis, including circulating cancer cells [Alix-Panabières et al., 2012], protein and mRNA biomarkers [Skog et al., 2008], and circulating tumour DNA (ctDNA) [Bettegowda et al., 2014]. Most encouragingly, a recent study achieved excellent results by sequencing mRNA derived from tumour-educated platelets, distinguishing cancer patients from healthy patients with 96% accuracy and differentiating between the six primary tumour types studied with 71% accuracy [Best et al., 2015]. Another prominent recent study used a combination of ctDNA and protein biomarkers to achieve a median sensitivity of 70% across eight common cancer types, with a specificity of 99% [Cohen et al., 2018]. GRAIL Inc., a spin-off from the sequencing technology company Illumina which has been the subject of a moderate amount of lay media attention, is currently running three studies into early detection of cancer by ctDNA. While they have not yet published any results in peer-reviewed journals, they have presented some promising initial results at a number of congresses [GRAIL Inc., 2019]. For example, at the 2018 American Society of Clinical Oncology (ASCO) Annual Meeting they reported sensitivities ranging from 38–51% at a specificity of 98% for detection of early-stage lung cancer.

There are two main problems with much of the current research in the field, when viewed with an eye to developing early tests for cancer. The first is that samples are not collected prospectively prior to diagnosis—cancer patients are recruited and samples collected after cancer has been clinically diagnosed (as in Cohen et al. [2018]

and Best et al. [2015]), which limits the ability to use these studies to assess a biomarker’s suitability as a screening test for use in patients whose cancer status is unknown. The second problem is the high cost of sequencing-based tests. Cohen et al. [2018] estimate their test to cost “less than \$500”, which presumably implies the cost is approximately \$500 (roughly £400).³ It has been reported that GRAIL are intending for their test to be priced similarly [Swanson, 2016]. This is significantly more than the cost of existing population-screening tests: the faecal occult blood (FOB) test for colorectal cancer, which in the UK is offered to everyone aged 60–74 every two years [NHS England, 2016], costs approximately £5 per test [NHS England, 2017]; the UK breast cancer screening programme is open to women aged 50–70 every three years, with an average cost of £13 per patient per year [Pharoah et al., 2013]; and the pap smear used in the UK cervical cancer screening programme costs £21.67 per test [Karnon et al., 2004] and is offered to every woman aged 25–64 either every three or five years, depending on their age.

Sequencing-based tests can be comparable in cost to current methods for definitively diagnosing cancer in patients presenting with clinical symptoms indicative of cancer (for example, a colonoscopy cost the National Health Service (NHS) £372 in 2017 [NHS England, 2017], and a transthoracic needle biopsy cost the NHS roughly £400 in 2015 [Cree, 2015]). However, even in this setting they encounter the problem of patient acceptance of the test as a definitive method of diagnosis. Patients tend to overestimate the accuracy of a physical diagnostic test (for example, endoscopy) and consequently have a high threshold for acceptance of tests based on blood, urine, or stool samples alone as a replacement for physical diagnosis [Yossepowitch et al., 2007].

1.3 Aims and scope

We set out to investigate whether modern machine learning techniques can be applied to VOC data as measured by FAIMS to produce a viable, low-cost, non-invasive cancer screening test. Recognising that taking VOCs from a promising biomarker to a clinically-usable test is likely beyond the scope of a single PhD thesis, as part of our work we developed a publicly available software package for the analysis of FAIMS data. This implements a reproducible end-to-end pipeline incorporating modern machine learning techniques and best practice for avoiding bias in reported

³As opposed to the cost being, for example, \$49, in which case one would assume that the phrase “less than \$50” would have been used.

results, which we hope will be useful for future research in the field.

We also investigated novel techniques for analysing FAIMS data, developing a novel Bayesian model which uses Gaussian processes to model data with structured correlations between variables and a high degree of redundancy in the observed data. We set out to explore the properties of this model through a number of computational experiments, including assessing its ability to model datasets with partially-observed data.

1.4 Methodology

1.4.1 Assessing classifier performance

To assess the performance of a predictive algorithm it is necessary to compare the output of the algorithm to known outputs for some test data, and a cornerstone of producing accurate estimates of performance is ensuring that the algorithm does not have access to the test set outputs, whether directly or indirectly. The worst case example of this principle being violated, where the training and test sets consist of identical data, can be easily seen to give essentially meaningless results through an illustrative example. Consider assessing the accuracy of a 1-nearest-neighbour classifier, which predicts the value of a test point by returning the output associated with the nearest point in the training data. If the training and test sets are identical, this classifier would appear to achieve perfect accuracy, which is unlikely to be borne out by more rigorous testing on unseen data.

Ideally, the sample size would be large enough to allow entirely distinct training and test sets to be defined, which makes proper application of this principle almost trivial. Unfortunately, generating data in a medical setting is an expensive business, and studies rarely have the luxury of setting sufficient data aside solely for testing and still retaining enough data for the study to have the requisite statistical power. However, a number of methods exist which allow one to make use of the the entire data set for both training and testing while minimizing positive bias through the disclosure of test set outputs to the training algorithm. These can be broadly described as resampling methods, where one uses the original dataset to generate some number of new pairs of training/test datasets, which are used for validation of the learning algorithm. Fundamental to the methods is that within each training/test pair, no data points are shared between both sets.

Resampling methods for validation of predictive models can be further classified into bootstrapping methods and cross-validation methods. We review the basic principles of these methods below.

Bootstrap resampling

Bootstrap resampling samples with replacement from the original data to generate a new training set of the same size as the original. The points which are not present in the training set are then used as the test set [Kung, 2014, p. 541]. If there are n points in the original data, the probability that a given point is not sampled (and therefore the proportion of points which do not appear in the training set) is

$$p(\text{point } i \text{ not sampled}) = \left(\frac{n-1}{n}\right)^n \quad (1.1)$$

which, rewritten as $(1 - 1/n)^n$, can be seen to converge to $\exp(-1) \cong 0.368$ as the sample size $n \rightarrow \infty$.⁴

Generally, one will undertake as many bootstraps as can be reasonably performed (within the limits of the available time and computational power) to obtain an estimate for the distribution of the performance metric being used and reduce the variance of the final estimate.

Cross-validation

Cross-validation is generally encountered as either k -fold cross-validation, or leave-one-out cross-validation (LOOCV). In k -fold cross-validation, each data point is assigned to one of k folds. Each fold is then used in turn as the test set, with the algorithm trained on the remaining out-of-fold points and again used to predict the left-out points in the test set. In LOOCV, each point is “left out” in turn, with the algorithm trained on the remaining points. This is then used to create a prediction for the left-out point. The process is repeated for every point in the data, and the predicted values are then used to calculate the performance metrics for the algorithm. LOOCV is equivalent to k -fold cross-validation with k set to the sample size of the original data. [Bishop, 2006, pp. 32–33; Kung, 2014, p. 539–540].

Performance metrics can be calculated in two ways from the results of k -fold cross-validation: either the predicted outputs can be aggregated into one set and

⁴ $\exp(-1)$ is a good approximation for the proportion of unsampled points even for small sample sizes. For example, at a sample size of $n = 10$, $(1 - 1/n)^n \cong 0.349$.

performance metrics calculated on that, or performance metrics can be calculated separately for each fold and averaged to find the performance metric for the entire data set [Forman and Scholz, 2010; Kung, 2014, pp. 539–540]. In this work, unless otherwise specified, we use k -fold cross-validation with $k = 10$ and performance metrics calculated on aggregated predictions across folds.

Pitfalls

Even if cross-validation is used to avoid bias, it is still possible to inadvertently and indirectly leak information about the test set to the classifier.⁵ An example we have encountered is the use of pre-processing steps outside of the cross-validation process which have access to class information, such as LDA, which attempts to find a low-dimensional representation of data while maximising the separation between classes in that representation. The LDA representation of the data is then split using cross-validation. Doing so will inadvertently give information about the test points' classes to the classifier, providing a significantly rosier outlook on the classifier's performance than might be achieved on truly unseen data. The obvious solution is to ensure that any process which uses class information occurs inside the cross validation, after the data has been split into folds. For example, in this case the LDA analysis should have been performed separately for each training set in the cross-validation.

1.4.2 Metrics

Classification algorithm performance is multifaceted, and different classifiers may be optimal in different contexts. A screening test will have significantly different performance characteristics to a diagnostic test. As such, the choice of metric we use to assess a classifier's performance requires careful thought. In interdisciplinary work, where the results of a statistical analysis are intended for consumption by clinicians as well as other statisticians, there is a second important consideration, that of metric acceptability and familiarity. Using a metric which is not familiar from the literature your readers are normally exposed to, even if that metric better captures the core of the problem you are assessing, will naturally lead to difficulties in communicating the results of your research. Here we review a number of metrics used in this work. In the discussion that follows we will assume that there are two

⁵The same is true for bootstrapping, or any other method of this sort. For simplicity, we will only refer to cross-validation for the remainder of this section, but the same principles apply to other similar techniques.

classes which points fall into, which can be described as the “positive” class and the “negative” class.

Sensitivity and specificity

Sensitivity (also known as recall) and specificity are defined as $TP/(TP + FN)$ and $TN/(TN + FP)$, respectively, where TP, TN, FP, and FN are the number of true positives, true negatives, false positives, and false negatives. They are widely understood measures of a test’s performance, but suffer from a number of drawbacks which makes them difficult to use for model selection in this setting. First, since they do not give a single metric and lack meaning when used in isolation,⁶ it is necessary to establish some method of jointly optimizing both metrics. Simple examples are fixing the specificity and optimizing the sensitivity, or vice versa, or by reference to some other outside constraint such as setting a minimum acceptable negative predictive value (NPV).

Positive and negative predictive value

Positive predictive value (PPV) (also known as precision) and NPV are the proportion of predicted positive (respectively negative) results which are true positives (or negatives). They are defined as

$$PPV = \frac{TP}{TP + FP} \quad (1.2)$$

$$NPV = \frac{TN}{TN + FN} \quad (1.3)$$

NPV is often used to describe screening tests, since it can be interpreted as the proportion of people who are negatively screened who are truly disease free, and so gives a measure of how many people who have the disease will be missed by the screening test.

Area under the ROC curve

Most classification algorithms we consider here do not output a hard classification for each data point, but will instead perform soft classification, where the classifier assigns a score to each point, with a higher score indicating that it is more likely to belong to the positive class. Defining a threshold for classification (where points with

⁶Any test can be made to have a sensitivity of 100% by predicting that every point tested is positive. Unfortunately (or fortunately, if you are a statistician who needs gainful employment), the specificity of this test is likely to be poor.

a score above the threshold are positive, and points below negative) creates a hard classifier, and the trained algorithm actually defines a spectrum of hard classifiers as the threshold sweeps through its possible range of values. While it is sometimes appropriate to fix the threshold, often we wish to consider the full range of classifiers defined by an algorithm, and for this we use the receiver operating characteristic (ROC) curve and the associated area under the curve (AUC). An example of a ROC curve can be seen in Figure 2.10.

For a given set of scores output from a soft classifier, the ROC curve plots the true positive rate (or sensitivity) against the false positive rate (or one minus the specificity) as the threshold varies across its allowed values [Krzanowski and Hand, 2009, p. 11]. The AUC, the area bounded by the ROC curve, the x -axis, and the line $x = 1$, is a measure of the classifier's performance across all possible thresholds, and is equivalent to the Mann-Whitney-Wilcoxon (MWW) U-statistic [Krzanowski and Hand, 2009, pp. 65–67], the statistic which underlies the MWW test, a robust non-parametric test of whether two samples are drawn from the same underlying distribution. This relationship can be exploited to calculate p-values for a given AUC, if required.

1.4.3 Classification models

In our work on FAIMS data we use a range of classification models intended to provide good coverage of different design principles in machine learning. The classification models we use are covered in more detail in Section 2.3.6, and we provide a brief overview here. The main models investigated were:

Random Forests Random Forests use an ensemble of decision trees to reduce the variance of the trees by averaging over their outputs. Generating the ensemble is a stochastic process, with two sources of variation. First, each tree is generated on a bootstrap resampling of the training data. Second, for each split only a random subset of the features are selected from (known as feature bagging).

Sparse logistic regression In sparse logistic regression, a penalty term is applied to the loss function for logistic regression which induces sparsity in the learned parameters. We use the elastic net penalty term. Elastic net is a linear combination of the lasso and ridge regression penalty terms, which correspond to normal and Laplace priors (respectively) being placed on the parameters.

Support vector machines Support vector machines (SVMs) attempt to find a maximally separating hyperplane between the classes in the training data. While this can be done in the observed data space, it is often beneficial to project the data into a high-dimensional space and find a maximally separating plane in this high-dimensional space. The kernel trick (see Section 3.2.2) allows this to be done implicitly.

Stochastic gradient boosting Gradient boosting attempts to iteratively improve its predictions by sequentially growing an ensemble of weak classifiers based on decision trees, at each stage attempting to fit the gradient of the loss function at the current predicted values and using this gradient to move the predictions closer to an optimum. Stochastic gradient boosting extends this by using a new bootstrap resample of the training data for each tree, as in Random Forests.

The models above were selected to represent the following broad categories of classifier algorithm:

- Decision trees;
- Bagging;
- Boosting;
- Kernel methods;
- Sparsity-inducing priors.

1.4.4 Gaussian processes

A Gaussian process (GP), broadly speaking, defines a probability distribution over functions. Formally, a Gaussian process is a set of random variables, any finite subset of which have a joint Gaussian distribution. This joint Gaussian distribution is defined by the GP's mean function $m(x)$ and its covariance function (or kernel function) $k(x, x')$, which are used to derive the mean and covariance matrix of a realisation of a GP for a given set of points. The covariance function plays a particularly important role, and is used to specify what class of functions a GP is a distribution over. Covariance functions exist which define GPs over linear, smooth, and periodic functions, as well as many other function classes. For example, covariance functions also exist for more exotic classes of function, such as the class of functions which can be produced by a multilayered neural network [Rasmussen and Williams, 2006].

Given a set of observed data, a GP can be used as a prior to encode the class of functions which we believe could have generated the data. The posterior distribution can then be used to make predictions about unobserved points. This is known as GP regression, and has a number of benefits over other non-linear regression techniques. Like many machine learning techniques, most covariance functions have a number of hyperparameters which need to be tuned to achieve good performance. GPs have a natural and principled method for fitting hyperparameters by optimizing the marginal likelihood of the model over the observed data. Also, since GP regression is a probabilistic method, predictions are provided with a variance giving the degree of confidence in the prediction. Finally, it is possible to craft (either manually or automatically) compound kernels which allow for flexible modelling of complex data structures, which we discuss in Chapter 5.

Gaussian processes are treated in more detail in Chapter 3.

1.4.5 Probabilistic PCA

Probabilistic PCA (PPCA) is a linear dimensionality reduction technique which sets PCA in a Bayesian probabilistic framework [Bishop, 1999]. In this formulation, one assumes that an observed data point \mathbf{x} is drawn from a multivariate normal distribution with variance $\sigma^2 \mathbf{I}$ around a linear transformation \mathbf{W} of the point's latent representation \mathbf{z} :

$$\mathbf{x} \mid \mathbf{z}, \mathbf{W}, \sigma \sim \mathcal{N}(\mathbf{W}\mathbf{z}, \sigma^2 \mathbf{I}) \quad (1.4)$$

One then set a prior $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ over \mathbf{z} , integrates \mathbf{z} out, and maximizes the likelihood of the resulting marginal distribution. Doing so, we find that the optimal \mathbf{W} agrees with that given by classical PCA.

Dual PPCA marginalizes out the loadings matrix \mathbf{W} rather than the latent variables \mathbf{z} , using a prior of $W_{ij} \sim \mathcal{N}(0, \alpha^2)$, to give a marginal distribution with the probability density function

$$p(\mathbf{X} \mid \mathbf{Z}, \sigma) = \frac{1}{(2\pi)^{dN/2} |\mathbf{K}|^{d/2}} \exp\left(-\frac{1}{2} \text{tr}(\mathbf{K}^{-1} \mathbf{X} \mathbf{X}^T)\right) \quad (1.5)$$

where $\mathbf{K} = \alpha^{-2} \mathbf{Z} \mathbf{Z}^T + \sigma^2 \mathbf{I}$ and \mathbf{Z} is the matrix of latent variables [Lawrence, 2004]. Again, optimizing \mathbf{Z} against the likelihood of the resulting marginal distribution gives a result which can be shown to be equivalent to classical PCA [Tipping, 2001].

Detailed derivations for PPCA and dual PPCA are given in Section 3.2.5.

1.4.6 The Gaussian process latent variable model (GPLVM)

Lawrence [2004] takes dual PPCA as a starting point, notes that the form of \mathbf{K} in equation 1.5 is that of a linear kernel (and that equation 1.5 is the marginal likelihood of the product of d GPs), and asks what happens if we replace \mathbf{K} with a non-linear kernel. The result is the Gaussian process latent variable model (GPLVM).

GPLVM assumes that observed data is a non-linear function of some latent representation, with the mapping functions from the latent space to the observed space drawn from a GP with kernel \mathbf{K} . To find the latent representation \mathbf{Z} for our data, we optimize \mathbf{Z} and the kernel hyperparameters to maximize the marginal likelihood of the model. For non-linear kernels there is no general analytic solution to this optimization problem, but we can attack the problem using standard iterative optimization techniques such as conjugate gradients [Møller, 1993] (as they do in Lawrence [2004]) or the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [Byrd et al., 1995].

1.4.7 Stochastic gradient descent

Gradient descent is an intuitive iterative optimization algorithm in which one seeks a minimum (or maximum) of a once-differentiable function by taking repeated small steps down (or up) the gradient until convergence is achieved. Stochastic gradient descent (SGD) introduces a random element to gradient descent by using a stochastic approximation to the gradient at each step [Bottou, 2010]. This has a number of benefits over exact gradient descent: first, if the gradient is expensive to compute exactly, stochastic gradient descent can make optimization feasible where exact gradient descent is not; second, if the function is not convex and has multiple local minima which are not globally optimal, the random nature of SGD can “bounce” the algorithm out of a local minimum and allow convergence to proceed towards a better solution.

SGD is not without its flaws. Setting an appropriate step size can be challenging, and often needs to be tuned to the specific problem. Too large a step size and the process will fail to converge correctly, with the next point selected frequently having a larger value than the previous point. Too small a step size and convergence will be too slow to complete in a reasonable time frame. Gradient descent and SGD also suffer from getting stuck in saddle points [Dauphin et al., 2014]. A number of

variants of SGD have been developed to address these and other issues. We go into some of these variants in more detail in Section 4.2.4.

1.5 Overview

Chapter 1 introduces the work.

Chapter 2 covers background information on VOCs, and our work on using VOC data for disease detection and diagnosis. We introduce our R package for reproducible analysis of FAIMS data, `FAIMSToolkit`, and review its functionality. Finally, we present a selection of studies in which we applied our package to the early detection of disease by urinary VOC analysis.

Chapter 3 covers background material on Gaussian processes and GPLVMs.

Chapter 4 introduces the theory of SGPLVMs, and covers our method for stochastic optimization of the likelihood of SGPLVM to make fitting the model computationally feasible. It also includes the results of a number of computational experiments in applying SGPLVM to both artificial and real data, and explores the properties of SGPLVM, including a comparison of its performance relative to GPLVM, its ability to fit a model to partially observed data, and its ability to visualise FAIMS data.

Chapter 5 presents our R package `gaussianProcess`, which fits Gaussian process regression models and performs automatic model selection by searching the space of compound kernels. It also reviews implementation details of our R package `GPLVM`, which implements a range of GPLVM variants, and also implements SGPLVM and the stochastic optimization method described in Chapter 4, as well as a number of SGPLVM variants.

Chapter 6 reflects on the results covered in this thesis.

Chapter 2

Volatile Organic Compounds: A Novel Biomarker for Low-cost Cancer Detection

2.1 Overview

2.1.1 Volatile Organic Compounds (VOCs)

VOCs represent a broad range of organic compounds. Informally, VOCs are organic compounds which evaporate at room temperature and atmospheric pressure. Formal definitions vary, but are broadly equivalent and generally use either the vapor pressure or boiling point of a substance under given conditions. For example, the International Union of Pure and Applied Chemistry (IUPAC) define a VOC as “any organic compound having, at 293.15 K, a vapor pressure of 0.01 kPa or more, or having a corresponding volatility under the particular condition of use” [Duffus et al., 2007], while the EU defines a VOC as “any organic compound having an initial boiling point less than or equal to 250 °C measured at a standard pressure of 101.3 kPa” [Council of the European Union, 2004].

For the purposes of early cancer detection (and more generally as a medical diagnostic tool), we are interested in VOCs which act as metabolites within the body. For cancer detection, this could either be VOCs produced by cells within a malignant or pre-malignant neoplasm which are not otherwise present in the metabolome of a healthy patient, or it could be a change in the concentration of one or more VOCs due to the either the neoplasm itself or the body’s response to it. As such, the use of VOCs for diagnostic purposes falls within the field of metabolomics. VOCs

are measurable in a wide range of human tissue sample mediums, including blood [O'Hara et al., 2009], urine [Hanai et al., 2012], saliva [Soini et al., 2010], and sweat [Dixon et al., 2007]. Endogenous VOCs are also measurable in exhaled air [Phillips et al., 1999], which offers a promising (although technically challenging [Alonso and Sanchez, 2013]) new medium for non-invasive medical testing.

A number of different analytic methods are available which measure VOCs. Typically, VOCs are measured in a gas-phase analyte drawn from the headspace of a liquid- or solid-phase sample. To aid VOC extraction the sample may be heated [Covington et al., 2015], or combined with an additive such as a salt [Jochmann et al., 2006]. For the detection of cancer using VOCs, a wide range of methods have been investigated: mass spectrometry (MS) [Wehinger et al., 2007]; GC-MS [Bajtarevic et al., 2009]; ion mobility spectrometry (IMS) [Westhoff et al., 2009]; FAIMS, a variant of IMS [Arasaradnam et al., 2014]; solid-state chemical sensors, also known as electronic noses or e-noses [Machado et al., 2005; Peng et al., 2010; Westenbrink et al., 2015]; colorimetric sensor arrays [Mazzone et al., 2007]; and specially-trained dogs [Cornu et al., 2011; Buszewski et al., 2012].

The work in this chapter is focussed on measuring VOCs using FAIMS, a form of IMS. In an IMS analysis, a gas-phase analyte is ionised and its composition determined using the characteristic electrical mobility of its constituent ions in an electric field. FAIMS exploits the fact that an ion's mobility in an electric field is a non-linear function of the field strength, and passes the ionised analyte through an oscillating electric field called the dispersion field, with a large difference in amplitude between the positive and negative parts of the cycle. Ions zigzag through the field, and ions with a non-zero net transverse motion over the full cycle will discharge themselves on the electrodes generating the field. Ions with zero net transverse motion pass through the device and discharge on sensors which record the positive and negative ion currents. A small "compensation voltage" can be applied across the electric field to adjust the net transverse drift of the ions. The dispersion field strength and compensation voltage are varied to identify the ion species present in the sample, with different ions able to traverse the field at different combinations of dispersion field strength and compensation voltage. The result of this for each polarity of ion can be visualised as an image, and each ion species traces out a plume in the image, as shown in Figure 2.1. In the text that follows, one "run" of the FAIMS is one full sweep of the dispersion field strength and compensation voltage, producing the data shown in Figure 2.1 for positive and negative ion counts. A

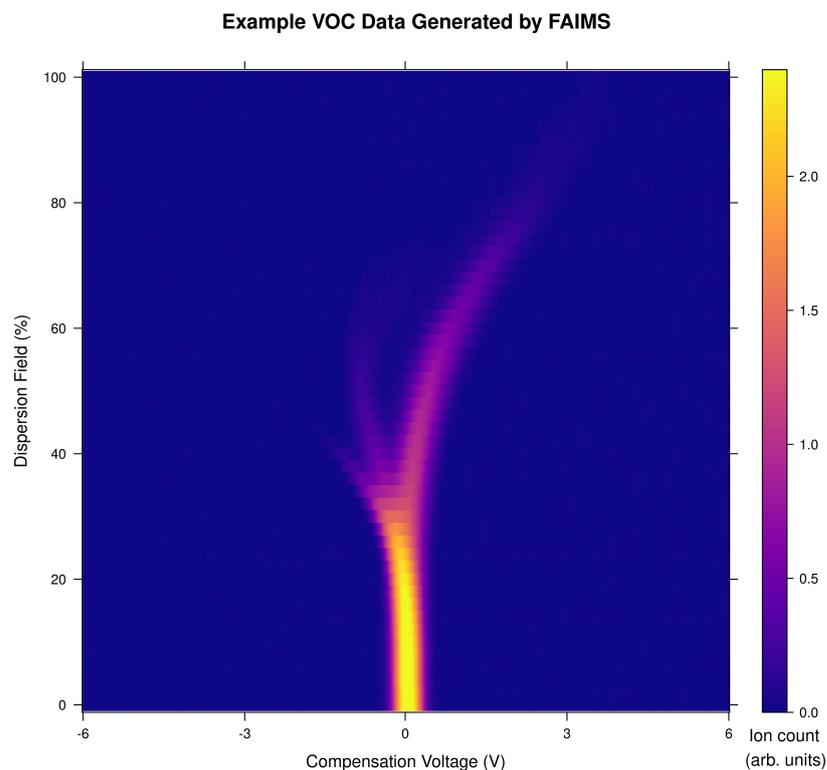


Figure 2.1: An example visualisation of the FAIMS data for the positive ion current for a single run. Different ion species in the sample produce distinct plumes, which branch off as the dispersion field strength increases.

FAIMS analysis of a sample will normally be composed of multiple runs to capture the changing VOC profile as the sample outgasses into the headspace.

There are two schools of thought on the application of VOCs to medical diagnosis. In one school of thought, it is beneficial to identify the specific VOCs characteristic of a disease process (as in Silva et al. [2011]; Hanai et al. [2012]), while in the other it is only deemed necessary to identify a pattern of change in the measurements made by a VOC-analysis device without attributing these changes to specific compounds (as in Arasaradnam et al. [2014]; Covington et al. [2013]; Westenbrink et al. [2015]). The benefit of the former approach is that identified compounds can be linked to specific metabolic processes, potentially giving greater insight into the disease processes involved. In the latter view, the identification of specific compounds is seen as secondary to the requirement that the test successfully differentiate between healthy and diseased patients, and the introduction of a mapping step from the raw data to specific VOCs risks discarding useful data about the metabolome which can be

exploited by modern machine learning techniques. Our work on using VOCs for the detection of cancer falls into the latter camp: we focus on identifying patterns in the data output by VOC-analysis devices which correspond to a patient’s disease status, without characterising the compounds which contribute to that pattern.

2.1.2 The “Two Week Wait” urgent care pathway

Many of the patients in the studies this chapter covers were recruited from the Two Week Wait (TWW) referral pathway at University Hospitals Coventry & Warwickshire (UHCW). Introduced in 2000 by the NHS Cancer Plan, the aim of the TWW pathway is to reduce variation in people’s experiences with cancer diagnosis and treatment across the UK (“ending the postcode lottery”), and to improve early detection and treatment of cancer, improving outcomes [Department of Health, 2000]. To this end, patients presenting with symptoms which indicate they may have cancer are urgently referred to the appropriate TWW pathway. National Institute for Health and Care Excellence (NICE) guidance sets out the symptoms and test results which should lead to a TWW referral from primary care [NICE, 2015]. These recommendations are based on the observation of sets of clinical symptoms which exceed a threshold of 3% PPV for cancer [NICE, 2015].¹ Once urgently referred, the target is for patients to receive an initial appointment with a consultant within fourteen days.² Should a decision to treat be made, treatment should begin within 62 days from the initial urgent referral.

2.1.3 Other studies on detecting colorectal cancer with urinary VOCs

VOCs have previously been investigated as a diagnostic tool for colorectal cancer. An excellent summary of the current literature has been produced by Di Lena et al. [2016], of which three studies involved urinary VOCs and are reviewed below.

In a prospective pilot study by Arasaradnam et al. [2014] of 83 patients with colorectal cancer and 50 healthy controls (confirmed by recent colonoscopy), VOCs measured by FAIMS were reported to have an accuracy of 74%, sensitivity of 88%, and specificity of 60% for predicting whether a patient had colorectal cancer. However, these results may overstate the effectiveness of urinary VOCs because of a flawed statistical analysis—cross-validation was used to estimate prediction accuracy, but

¹Except for children and young people, for whom recommendations were made at below 3% PPV, with no explicit threshold set [NICE, 2015].

²Hence the name Two Week Wait.

feature selection using the class of the samples (cancer or not-cancer) was performed before the cross-validation took place, thereby leaking information about the class of the held-out data into the model. More detail on the statistical analysis method can be found in Covington et al. [2013].

A second study by the same authors used a bespoke electronic nose device to measure the urinary VOCs of 39 patients with colorectal cancer, 35 with irritable bowel syndrome (IBS), and 18 healthy controls, reporting a sensitivity and specificity of 78% and 79% respectively for distinguishing IBS from colorectal cancer [Westenbrink et al., 2015]. Unfortunately the statistical analysis in this study suffers from the same problem as Arasaradnam et al. [2014], with feature extraction using class information being performed using LDA before cross-validation takes place, biasing the estimated performance of the classifier.

Silva et al. [2011] used a gas chromatograph quadrupole mass spectrometer (GC-qMS) to identify VOCs in the urine of 33 cancer patients (14 leukaemia, 12 colorectal, and 7 lymphoma) and 21 healthy controls. They identified 82 compounds, of which 16 were found to have a statistically significant difference between the cancer and control groups in a one-way ANOVA analysis at a significance of 5%.³ The paper reports good qualitative separation of the four groups (three cancers and controls) using PCA after restricting to the 16 compounds found to be significant, however as with other papers in the field this is not properly validated using either hold-out data or cross-validation. In addition, the healthy controls were drawn not from a population of symptomatic patients who on investigation did not have cancer, but from healthy volunteers with no history of cancer, limiting the relevance of this result when considering evidence for VOCs as a clinical test for when colorectal cancer is suspected. Even so, the identification of 16 urinary VOCs characteristic of cancer is encouraging for the utility of urinary VOCs as a component of a population-level cancer screening test.

2.2 Challenges of working with FAIMS data

There are a number of challenges to analysing FAIMS data which we have encountered in the course of our work. Some, such as detecting failed samples and assigning autosampled data batches to samples, are mechanical problems in the laboratory

³Effectively a t-test at a 5% significance level, since one-way ANOVA for two factors is equivalent to Student's t-test.

which we can resolve using software. Others, such as the high-dimensional nature of the data, are inherent to FAIMS data and require careful application of statistical techniques. Finally, the sensitivity of FAIMS as a detection instrument introduces issues which need to be taken into account at the time of study design, and cannot be resolved once the data have been generated.⁴

2.2.1 Failed samples and outlier detection

Infrequently, samples will fail to run successfully, without any explicit indication being provided in the data or metadata generated for the sample. There are a number of possible causes of this which we have encountered, although this list is likely not exhaustive:

- Operator error - mistakenly running a blank (empty or water-filled) vial instead of the intended sample.
- When using manual sample introduction using an ATLAS (Owlstone, UK) sampling system, failing to properly secure the sample container, allowing the ingress of un-scrubbed environmental air into the FAIMS instrument.
- When using sample introduction by autosampler, failure of the autosampler needle to pierce the septum of the sample's vial.
- Loss of flow rate during sample analysis, for instance due to loss of pressure in the lab's clean air supply.

Luckily, flow rate is reported in the metadata for a sample. A low flow rate (below 80% of the target flow rate) at some point in the analysis provides a sufficient but not necessary condition for detecting a failed run, allowing definitive identification of many failed runs. Figure 2.2 shows some examples of FAIMS data with low flow rates. Prior to this approach, failed runs were detected using outlier detection and manual inspection, which would fail to detect some runs with low flow rates.

Unfortunately, not all failed samples are so easily identified. In these cases, it is necessary to rely on outlier detection, visual inspection, and familiarity with FAIMS data to determine whether a sample should be excluded. To detect outliers we calculated the mean distance to each point's k nearest neighbours (k-NN), as described in Angiulli and Pizzuti [2002], although our algorithm used a nested loop

⁴Much to the chagrin of one group of collaborators who approached us to analyse the data from their study.

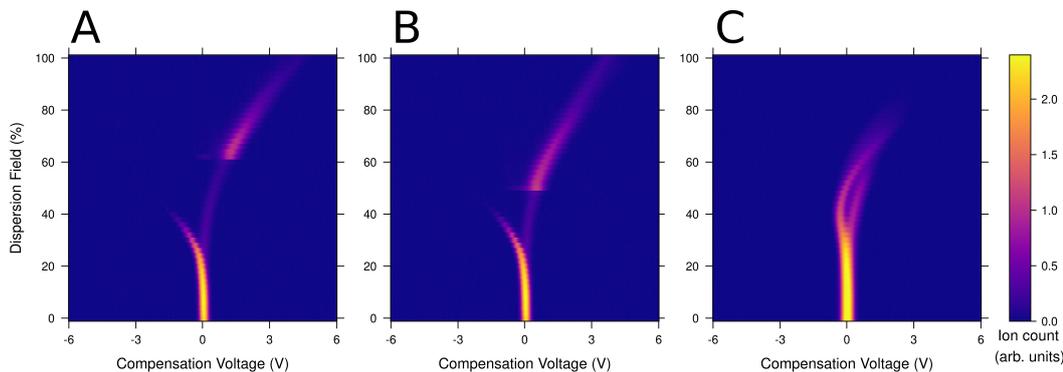


Figure 2.2: Examples of FAIMS samples which have low flow rates reported in their associated metadata. Flow rate issues are evident in **A** and **B** from visual inspection by the discontinuity in the plumes, but the quality issues with **C** are less apparent visually because the flow rate change occurs outside of the visible plumes. Previously data quality was assessed visually in the lab and **C** could therefore have been included in the statistical analysis, despite not having run correctly.

to calculate the mean k-NN distance in $\mathcal{O}(d \cdot n^2)$ time (where d is the dimension and n is the number of data points) rather than the faster and more sophisticated approach described in Angiulli and Pizzuti [2002], which scales linearly with the number of data points and the dimension of the data. Nonetheless, performance was acceptable for the data sets we worked with, and the output of both methods is equivalent.

Outlier detection by mean k-NN distance is simple to implement, easy to describe conceptually to collaborators from other disciplines, and has been found to perform comparably to other outlier detection algorithms [Campos et al., 2016]. A significant point in its favour is that its performance is relatively insensitive to the choice of the parameter k , compared to other outlier detection algorithms which require more careful parameter selection for optimal performance [Campos et al., 2016].

For our purposes we did not set a pre-defined number of outliers, but used a plot of two sets of k-NN mean distances (for example, $k = 2$ against $k = 10$) to identify possible outliers, which were then visually inspected to determine if they should be excluded. Figure 2.3 shows such a plot for the anastomotic leakage pilot study discussed in Section 2.4.4, along with some examples of outlier and non-outlier results.

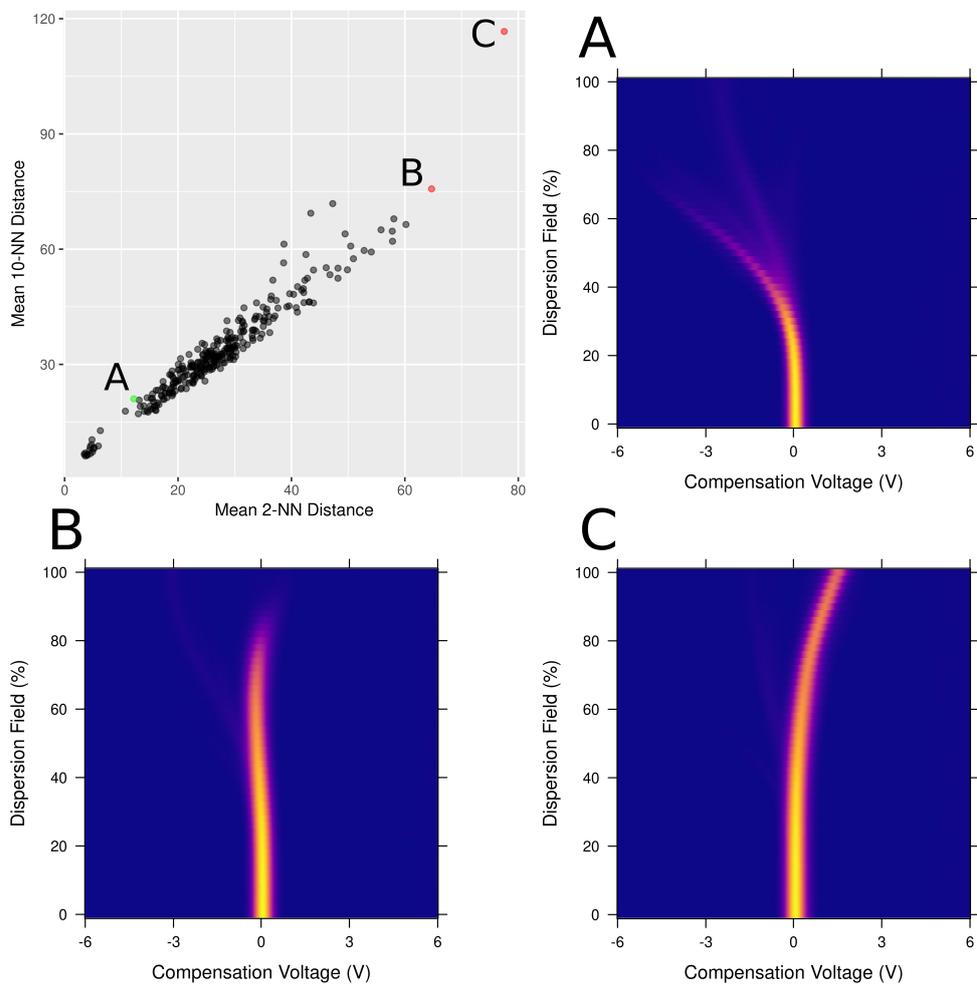


Figure 2.3: A plot of the mean k -NN distance ($k = 2$ against $k = 10$) for the data in the anastomotic leakage pilot study (*top-left*). Visualisation of the FAIMS data from the the highlighted samples are shown in the other four plots - **A** shows a non-outlier sample, while **B** and **C** show outliers with plumes which differ markedly from what is expected for these samples. Both outliers shown were excluded from the study.

2.2.2 Assigning autosampled data to sample IDs

Although the use of an autosampler improves reproducibility and reduces FAIMS analysis failure due to operator error, it also introduced a new source of error. In the autosampler/FAIMS configuration in use at our primary collaborator’s lab at Warwick University, there is no interface between the autosampler and the FAIMS instrument. Instead, the FAIMS takes continuous measurements, and the autosampler introduces samples into the FAIMS at regular intervals. Because the start of the FAIMS runs and the beginning of the autosampler introducing a sample are not synchronised, it is almost always the case that the first and last runs of a sample will only partially be measuring the sample headspace, and also that FAIMS runs are not labelled with the sample IDs.

Initially, FAIMS data was annotated manually by the lab technician visually inspecting the FAIMS output. Samples were first classified as either sample data or a between-sample run to be discarded, and then with the sample ID based on the order in which samples were placed in the autosampler. This process was time-intensive, with a batch of samples requiring multiple days of work just for manual annotation, and was also error-prone. Frequently runs which should have been classified as between-sample runs were incorrectly included with sample data.

To resolve this, we developed a piece of software to automatically identify in- and between-sample runs and assign in-sample runs to the correct sample ID, exploiting the fact that the flow rate was reduced when the autosampler was not introducing a sample. Suspected failed samples are reported for manual confirmation and correction. The documentation can be found in Appendix B, and the source code is available online at https://github.com/mattdneal/faims_autosampler_splitter. Figure 2.4 shows the output for identifying failed samples.

2.2.3 High-dimensional data

The data generated by FAIMS analysis are high-dimensional, normally on the order of 10,000–100,000 variables. For each run, the dispersion field strength takes 51 values between 0% and 100% field strength, and for each setting of the dispersion field the compensation voltage takes 512 values between -6 V and 6 V, yielding 26,112 variables per ion channel per run for a total of 52,224 variables per run. In most studies multiple runs are performed, with some studies performing up to ten runs for a total of 522,240 variables per sample. FAIMS is not alone in this regard amongst VOC analysis instruments—GC-IMS (Impex Diagnostics Ltd, UK),

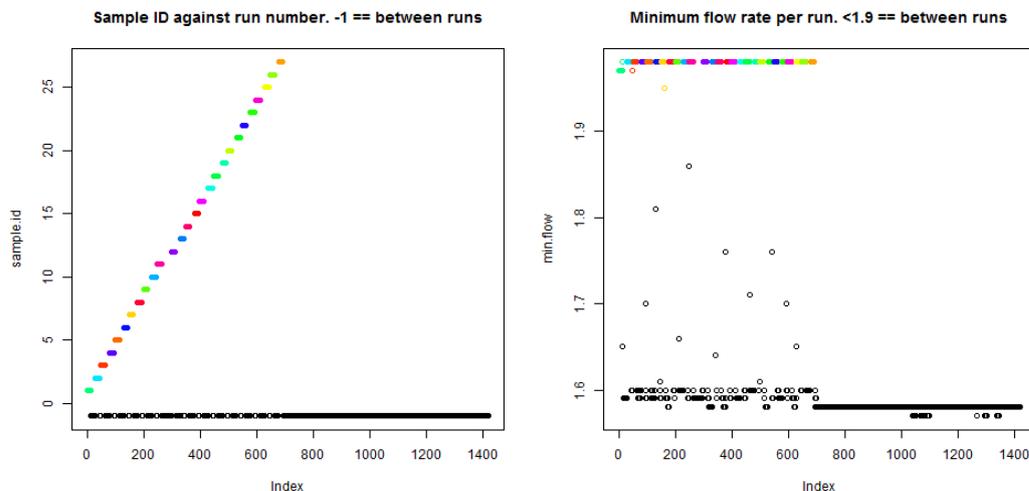


Figure 2.4: Example output from the tool we developed to automatically identify in-sample runs and assign sample IDs to autosampled FAIMS data. A failed run is visible as a gap in the series of coloured sections on the right-hand plot, and as a distinct step in the in-sample runs in the left-hand plot (between sample IDs 11 and 12), suggesting that the 12th sample in the batch failed to run correctly.

another promising tool for low-cost measurement of VOCs, produces data with 5,143,500 variables per sample.

Coupled with the characteristically small sample sizes associated with most early cancer detection studies, the high-dimensional nature of the data presents a number of problems, from more mundane implementation problems (e.g. can we hold the entire data set in memory) to the thornier issue of dimensionality reduction and feature extraction without overfitting or losing information.

Wavelet transforms

The standard analysis pipeline for FAIMS data uses a wavelet transform to concentrate the relevant macroscopic features of the FAIMS data (the plumes) into fewer variables, which could then be identified by a feature selection algorithm. To reduce the dimensionality of the data, we extended this by discarding the first level of wavelet coefficients, corresponding to the highest frequency components of the data. Since the signal in the data is highly correlated between neighbouring variables (those with adjacent dispersion field and compensation voltage values), the highest frequency components correspond to noise, and their removal has not been observed

to affect classifier performance. By removing the highest frequency components the number of variables to consider is reduced by half.

Noise reduction

In addition to this, a large portion of the FAIMS output (60–75%) contains no useful information since it does not contain a VOC plume. We implemented two methods for identifying variables which did not contain VOC plumes,⁵ both of which produce similar results. If wavelet transformation is to be undertaken, noise variables can either be set to their median value or a noise threshold can be calculated and subtracted from every variable, with variables below the threshold set to zero. This preserves local smoothness and prevents sudden changes in value if noisy variables are simply zeroed out. If the data are not going to be wavelet transformed then noise variables can be excluded from the analysis entirely.

The more straightforward method identifies variables whose standard deviation is below a given threshold, which can be selected by using the elbow of a cumulative distribution graph of the standard deviation across all variables (see Figure 2.5 for an example). The second method exploits the fact that signals in the data are highly correlated locally between adjacent variables, while noise is locally uncorrelated, by scoring each variable based on the mean correlation between a variable and its immediate neighbours. Again, a threshold is then selected based on a cumulative distribution plot for this score across all variables. Figure 2.5 shows a comparison of the results for the two methods for one dataset. Generally, the standard deviation method was preferred for its simplicity and more conservative identification of noise variables.

2.2.4 Study design considerations

The sensitivity of FAIMS analysis to systematic effects can exacerbate problems with a study’s design, to the point of rendering the gathered data useless. Specifically, FAIMS analysis is very susceptible to batch effects due to VOC contamination of sample batches which are correlated with the outcome of interest. For example, if samples for disease-positive patients are processed by one researcher, and samples for disease-negative patients are processed by a different researcher, it is possible for the samples’ VOC profile to be affected by the VOC profile of the researcher,

⁵For implementation details, please see the source code for the functions `findNoiseFaimsData.sd` and `findNoiseFaimsData.localCorr`. The source code is available online at <https://github.com/matt dneal/FAIMSToolkit>

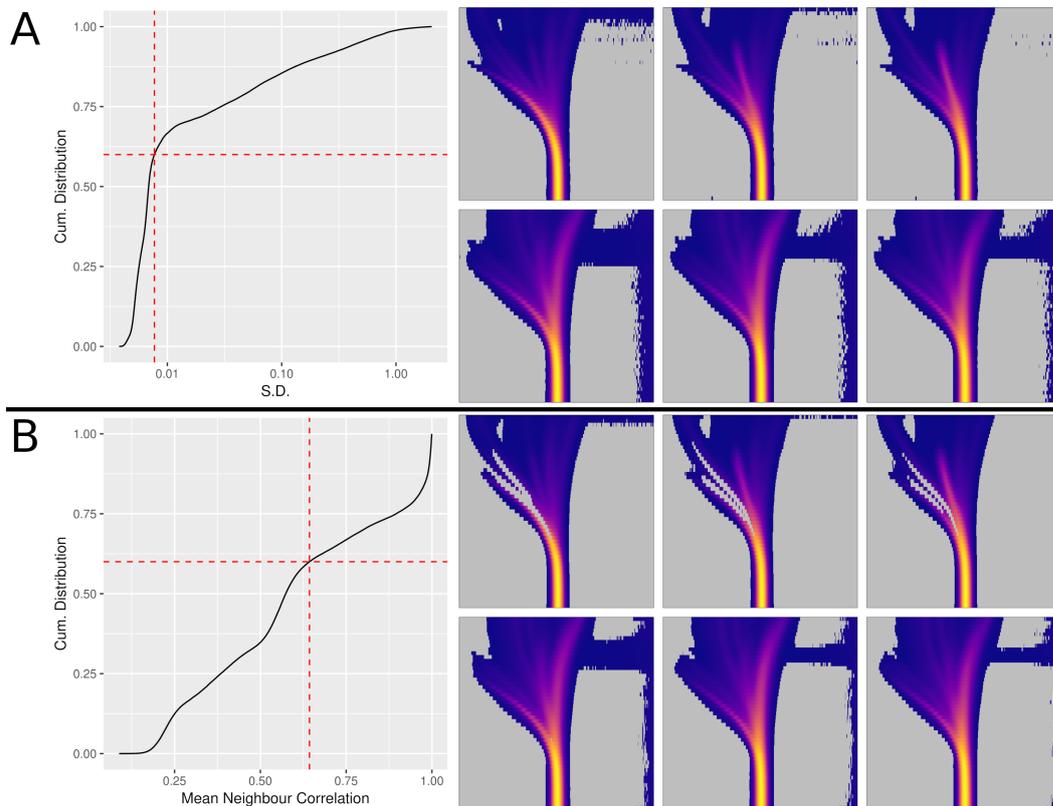


Figure 2.5: Example results for identifying 60% of variables as noise using (A) standard deviation and (B) mean correlation to neighbouring pixels. In each figure, the plot on the left shows the cumulative distribution graphs for the two metrics, with the dashed lines showing the noise threshold. The six right-most plots in each figure display the resulting data, showing positive and negative ion counts for three runs. Pixels in grey correspond to variables excluded as noise. Included variables are coloured based on the mean value of that variable.

and as such one constructs a method of differentiating between the deodorants of the two researchers, rather than detecting the disease of interest. Exactly such a situation occurred in the study discussed in Section 2.4.3. Other possible sources of error are: analysing samples in batches of one class (disease-positive in one batch, and disease-negative in another); collecting samples for different classes in different locations or using different personnel; and storing samples from different classes for significantly different lengths of time. There are to our knowledge no published examples of such, perhaps due to publication bias against null results.

In general, it is necessary to carefully randomise every step of a study, including sample collection, storage, and analysis. In situations where class sizes are very unbalanced, as is often the case with cancer detection studies, one should also consider stratifying samples by class where possible to avoid batches of samples containing only one class.

2.3 Methods

To facilitate analysis of FAIMS data, we developed an R [R Core Team, 2018a] package called `FAIMSToolkit` (totalling 2230 lines of R code as of version 0.2.15) containing functions for reading, processing and classifying FAIMS data. The source code for the package is available online at <https://github.com/mattdneal/FAIMSToolkit>, and the documentation can be found in Appendix C. Figure 2.6 shows the typical software flow when using `FAIMSToolkit` to generate a classification model for FAIMS data, and Figure 2.7 shows the typical data flow along with some of the configurable steps which can be included.

2.3.1 Managing FAIMS data

Since each variable in FAIMS data is associated with a compensation voltage, dispersion field strength, run number, and ion polarity, the data for a single sample can be considered as a four-dimensional array, rather than as a vector, and the data for a set of samples can be considered to be a five-dimensional array rather than a matrix. However, for many functions it is more convenient to work with the data as a vector or matrix. To address this without losing track of the array dimensions, the package manages FAIMS data using R's S3 objects, which are described in section 5 of the R Language Definition [R Core Team, 2018b]. Alongside the data in matrix form, the array dimensions are stored in the object to allow an array representation to be recovered as needed. In addition, a vector giving the minimum flow rate for each sample in the dataset is stored for addressing the issues discussed in Section 2.2.1. `FAIMSObjectFactory` is provided for instantiating FAIMS objects, the utility function `deleteFAIMSSample` facilitates the deletion of samples from the dataset, and `convertToArray` will convert a FAIMS dataset stored in matrix format to array format.

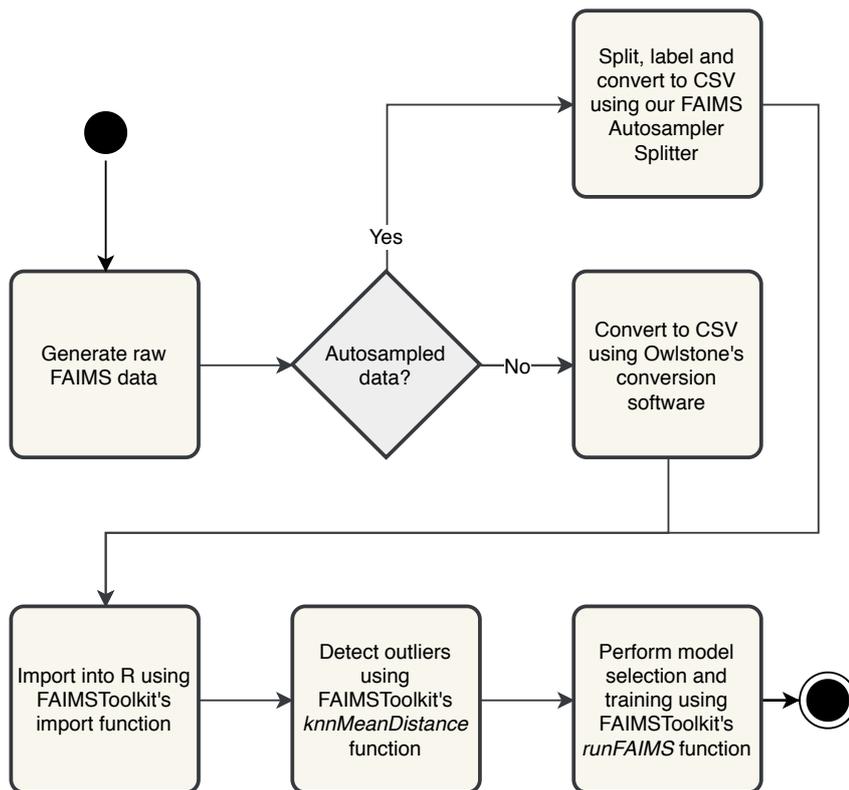


Figure 2.6: UML 2 [Object Management Group, 2017] activity diagram showing the typical use of our FAIMS software, from generating the raw FAIMS data files to producing a trained model. A number of different data pre-processing and model training options are available in `FAIMSToolkit`—these are discussed in more detail in the text, and some are also shown in Figure 2.7. This diagram shows the software activity flow if the default options are used.

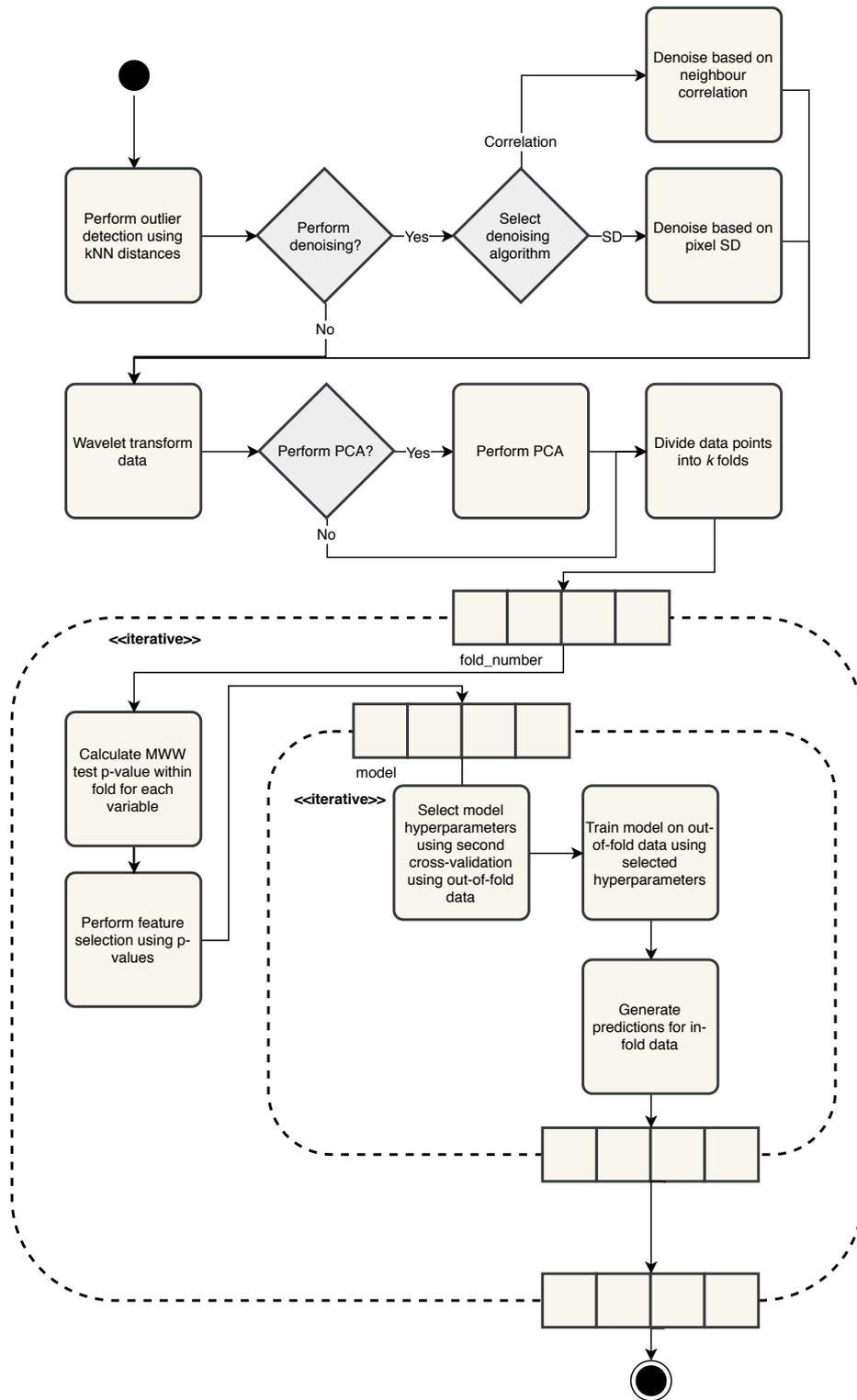


Figure 2.7: UML 2 [Object Management Group, 2017] activity diagram showing the typical data flow when using our FAIMS software.

2.3.2 Importing FAIMS data into R

`FAIMSToolkit` contains a number of functions for importing FAIMS data into R. The function `ReadInFaimsDirectoriesMultiFile` imports manually sampled data, while `ReadInFaimsDirectoriesAutosampler` imports autosampled data. Both automatically check for flow rate issues on import, combine data from multiple runs per sample into one entry per sample, and check that the number of runs per sample is as expected.

2.3.3 Outlier detection, denoising & data transformation

`knnMeanDistance` implements k-NN outlier detection, as described in Section 2.2.1. `findNoiseFaimsData.sd` and `findNoiseFaimsData.localCorr` identify noisy variables by standard deviation and local correlation respectively, as described in Section 2.2.3. `WaveletTransform` and `WaveletTransform_2D` implement one- and two-dimensional wavelet transformations, respectively, again as described in Section 2.2.3.

2.3.4 Visualisations of FAIMS data

Two functions are included for visualising FAIMS data using two-dimensional plots. Both functions use the R package `viridis` [Garnier, 2018] for the ion count colour maps, if it is installed. `viridis` provides perceptually-uniform colour maps which are accessible to people with the most common forms of colour-blindness and also transfer well to greyscale [Nuñez et al., 2018], providing a number of advantages over the base-R colour maps. `plotFAIMSdata` takes a FAIMS object and sample IDs and plots those samples' data with unlabelled axes and no colour map legend, for quickly visualising many samples' data. The colour map for the plots is derived from the full dataset rather than individual samples, allowing meaningful comparison between plots within one dataset. An example of the output from this function can be seen in Figure 2.8. `prettyFAIMSPlot` plots a single run and ion polarity for one sample, with labelled axes and an ion current colour map legend, such as the one shown in Figure 2.1.

2.3.5 Classifier training and validation

`FAIMSToolkit` incorporates a number of methods for feature extraction and classifier creation. To streamline the process of analysing a set of FAIMS data, `runFAIMS` is provided as a convenience function which wraps the most common parts of the analysis pipeline, allowing one to provide a list of classifier models to construct, whether PCA

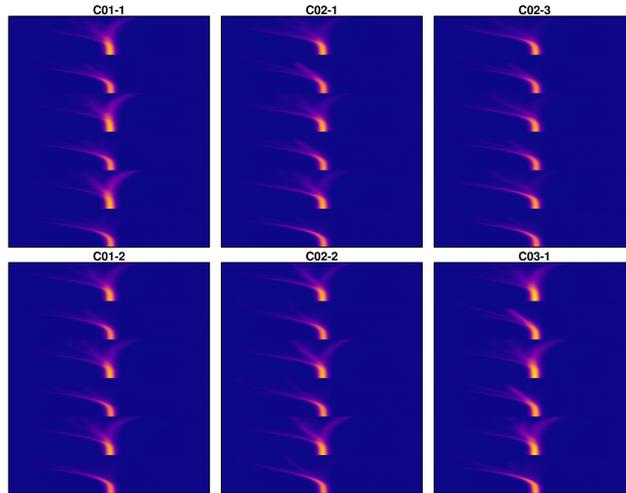


Figure 2.8: Example output from `plotFAIMSdata` for six samples. Plot titles are the sample IDs.

should be applied before classification, and how variable selection should occur. It then performs a 10-fold cross-validation with in-fold feature extraction and model parameter tuning. Details of this process are given below.

Feature Selection

Within each fold, features are selected from the training set using the MWW test [Mann and Whitney, 1947], a non-parametric test for whether two independent samples are drawn from the same distribution, by comparing the distributions of each variable for the two classes with the null hypothesis that the underlying distributions are identical.

Two methods of feature selection are available in the package. The first is to simply specify the number n of features to retain, in which case the n features with the lowest p-values under the MWW test are retained and used to train the model. The second uses a multiple hypothesis test correction technique called sequential goodness of fit (SGoF) [Carvajal-Rodríguez et al., 2009], which given a significance level α models whether the features are found to be significant as a binomial distribution with $p = \alpha$ under the null hypothesis that no feature are truly significant. It then excludes a number of features equal to the $(1 - \alpha)$ quantile of the null hypothesis binomial distribution, q . If s features are found to be significant under MWW, SGoF would therefore accept the top $s - q$ of those features (by p-value) as truly significant, discarding the remaining variables.

Principal Component Analysis

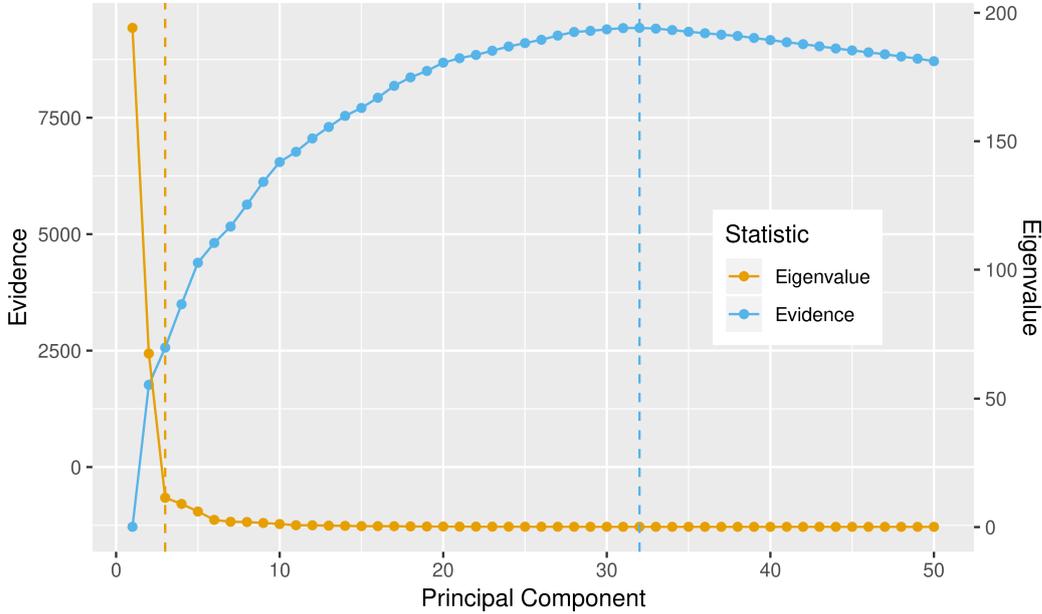


Figure 2.9: Plot of (in blue) the number of principal components retained against the evidence (as described in the text), and (in orange) a scree plot of the eigenvalue associated with each principal component, for a principal component analysis of one fold in the study discussed in section 2.4.4. Dashed lines show the number retained by each method, with 33 principal components retained using the Bayesian model selection method, compared to 3 for the scree test.

After feature selection, `FAIMSToolkit` can optionally perform PCA on the selected features to reduce the dimensionality of the data. We implemented a Bayesian model selection method to determine the number of principal components to retain, as described in Minka [2001]. In brief, this uses the probabilistic PCA model [Tipping and Bishop, 1999] to assess the evidence⁶ for each value k of principal components to retain. To do this we calculate Laplace’s approximation to the evidence for each value of k and select the k for which the evidence is highest. This method has been shown to outperform other methods of automatic dimensionality selection for PCA when attempting to recover the true latent dimension of a data set [Minka, 2001].

⁶Evidence in the sense of $p(\text{Data} \mid \text{Model}) = \int_{\theta} p(\text{Data} \mid \theta)p(\theta \mid \text{Model}) d\theta$, where θ represents the unknown parameter values in the model.

Model hyperparameter tuning

Almost all machine learning algorithms have some number of hyperparameters, and in general the hyperparameter values which achieve optimum performance on one problem will not be optimal for a different problem. As such, for a given classifier it is generally necessary to undertake some sort of search of hyperparameter space to identify hyperparameter values which, if not optimal, at least avoid exceptionally poor performance. This process is known as model tuning. The importance of this process for classifier performance varies widely between classifiers [Probst et al., 2018]. Some classifiers, such as SVMs [Suykens and Vandewalle, 1999] and XGBoost [Chen and Guestrin, 2016], require careful tuning to achieve good performance, while others such as k-NN and Random Forests [Breiman, 2001] are relatively insensitive to hyperparameter tuning.

Hyperparameter tuning is not without its pitfalls. If one were to naïvely attempt to improve classifier performance by performing a fine-grained and computationally expensive search of hyperparameter space for optimum values, one runs the risk that the resulting classifier will generalise poorly due to overfitting of the training data. To mitigate this, for each fold in the 10-fold cross-validation, we perform a 5x2 cross-validation [Dietterich, 1998] to select model hyperparameters—a 2-fold cross-validation on the training data, repeated 5 times. Within each fold we perform a grid search with five values per hyperparameter.

We used the R package `caret` [Kuhn et al., 2018], a remarkably useful package which streamlines the process of model tuning and training. Among other things, it provides a wrapper function which integrates many R packages, providing a consistent interface for training over two hundred classification and regression models, and can be configured to perform 5x2 cross-validation for hyperparameter selection.

Classification models

Thanks to the use of `caret`, `FAIMSToolkit` can be used to assess any of the available classifiers provided by `caret`.⁷ The default models assessed by `runFAIMS` have been chosen to cover a wide range of classes of classification algorithms. They are: Random Forests [Breiman, 2001], provided by the `randomForest` package [Liaw and Wiener, 2002]; sparse logistic regression, provided by the `glmnet` package [Friedman et al., 2010]; RBF and linear SVM [Cortes and Vapnik, 1995], provided by the `kernlab`

⁷A full list can be found at <https://topepo.github.io/caret/available-models.html>, which as of September 2018 contained 237 classification and regression models.

package [Karatzoglou et al., 2004]; stochastic gradient boosting [Friedman, 2002], provided by the `gbm` package [Ridgeway, 2017]; and logistic regression, provided by the `glm` function in base R [R Core Team, 2018a]. More details on the workings of these models can be found in section 2.3.6.

Analysis progress feedback

Analysis of FAIMS data can take several hours to run. To provide feedback on overall progress while leaving the console free for more detailed debugging information, `FAIMSToolkit` creates a separate progress bar window using Tcl/Tk [Tcl Core Team, 2018], a cross-platform graphical user interface (GUI) framework which is embedded in most standard R distributions and can be accessed using the `tcltk` package [R Core Team, 2018a].

Overfitting detection

A danger when developing machine learning techniques is that increasingly complex analysis pipelines increase the possibility of overfitting. A similar effect can also be caused by programming errors inadvertently leaking the classes of the test data to the algorithm during the training phase. To detect this, `FAIMSToolkit` automatically replicates all analyses performed using it with the classes permuted randomly between samples. The expectation in this case is obviously a null result. If the result of the “scrambled” analysis is better than would be expected by chance, the results of the original analysis should be considered highly suspect, and the code should be debugged to identify the source of the overfitting.

2.3.6 Classification models

Random Forests

Random Forests, introduced by Breiman [2001], uses an ensemble of decision trees to reduce the variance of the underlying classifiers by averaging over their outputs. The decision trees are grown using CART methodology [Breiman, 2017]. Random Forests uses two methods to introduce variation into the population of trees it averages over:

Bagging For each tree, Random Forests performs a bootstrap resampling of the training examples. That is, for a set of n training examples, Random Forests trains each tree on n examples sampled with replacement from the original training set, with the resampling performed separately for each tree.

Random subset of features When training a tree, the split variables is chosen from a random subset of the features. This is intended to reduce the correlation between trees in the ensemble. This technique is also called feature bagging, attribute bagging, or the random subspace method.

Random forests are versatile classifiers with a number of advantages. They inherit the ability of decision trees to classify based on both continuous and categorical features; they are invariant to scaling of the features; they have few parameters and perform well with little or no hyper-parameter tuning; and they achieve a good level of performance in many classification tasks [Caruana and Niculescu-Mizil, 2006].

Sparse logistic regression

In sparse logistic regression a penalty term is applied to the standard logistic regression model to induce sparsity of the regression coefficients. The R package `glmnet` implements the elastic net method. The elastic net penalty term is a linear combination of the lasso and ridge regression penalty terms. The loss function for the model is:

$$L(X; \beta) = \|y - \sigma(X\beta)\|_2^2 + \lambda \left(\frac{1 - \alpha}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right) \quad (2.1)$$

where

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad (2.2)$$

is applied element-wise to vectors. Ridge and lasso regression occur as special cases of elastic net regression when α equals zero and one, respectively.

Support vector machines

SVMs find a maximally separating hyperplane between two classes by maximising the orthogonal distances between the hyperplane and the nearest points in each class.⁸ The constraint that the hyperplane must separate the data is known as hard-margin SVM. In cases where the classes are not linearly separable, soft-margin SVM may be used, where the constraint that the hyperplane must be separating is softened to a loss function which penalises planes with misclassified points. Even in the case of linearly-separable data, soft-margin SVM can guard against overfitting by allowing outliers to be misclassified for better generalization overall.

⁸The hyperplane therefore relies only on these points, which are known as support vectors, from which the name support vector machine is derived.

Since the operations required to optimize an SVM model can be expressed entirely using dot products, the kernel trick⁹ may be used to implicitly project the problem into a higher-dimensional feature space. This allows for non-linear decision boundaries and significantly increases the utility of SVMs.

Although SVMs can provide good accuracy [Caruana and Niculescu-Mizil, 2006], they have a number of drawbacks in practice. First, since SVMs use a geometric model for classifying the data, they are sensitive to the scaling of the features. Second, for non-linear SVM performance is generally dependent on good kernel hyperparameter selection, which can be computationally expensive.

Stochastic gradient boosting

Boosting is a technique where an ensemble of weak classifiers, whose performance only slightly exceeds guessing at random, are combined to produce a strong classifier. Stochastic gradient boosting [Friedman, 2002] is an extension which adds bagging (as in Random Forests) to Friedman's gradient boosting machine [Friedman, 2001]. The gradient boosting machine uses decision trees with a small maximum depth K to iteratively estimate the gradient of the loss function L with respect to the current estimate \hat{f} for the target function f . \hat{f} is then updated by adding the estimate for the gradient multiplied by a step size which minimises the loss function.

Stochastic gradient boosting adapts this in two ways. First, the step size (or learning rate λ) is defined in advance. Second, the tree used to estimate the gradient of the loss function is fit on a bootstrap sample of the training data to introduce stochasticity into the algorithm.

2.4 Experimental results

In this section we review a selection of VOC studies in which we have been involved. The studies covered are:

- A study of 526 patients in the Two Week Wait urgent care pathway for colorectal cancer, comparing the use of urinary VOCs to two stool biomarkers currently in clinical use for the diagnosis of colorectal cancer and other gastrointestinal diseases. In this study we present a novel correction for sensitivity and specificity of screening tests based on test compliance rate.

⁹See section 3.2.2

- A study investigating urinary VOCs as a screening test for colorectal cancer, the cohort for which comprised 127 patients with a positive FOB result attending the bowel cancer screening nurse clinic at UHCW.
- A study investigating the use of urinary VOCs to discriminate between colorectal cancer and IBS, which had a null result due to sensor drift and led to the development of a novel test for FAIMS sensor drift.
- A study into the use of urinary VOCs as a screening test for pre-symptomatic detection of anastomotic leakage following surgical resection of gastrointestinal cancers, involving 63 patients at VU University Medical Center (VUmc).
- A second study on the detection of anastomotic leakage by urinary VOCs following major colorectal surgery. This study was a multi-centre cross-sectional cohort study of 49 patients drawn from patients attending five hospitals.

2.4.1 UHCW Two Week Wait colorectal cancer pathway

In collaboration with UHCW, we investigated the use of urinary VOCs as either a supplement to or replacement for two non-invasive stool tests for the diagnosis of colorectal cancer and adenoma in the urgent TWW clinical pathway. Details of the study not covered here can be found in the published paper [Widlak et al., 2018].

Biomarkers investigated

Three biomarkers were considered in the study. Two stool biomarkers were studied: faecal haemoglobin (F-Hb) (also referred to as FOB), measured by faecal immunochemical tests (FIT), which detects blood in the stool [Mowat et al., 2016]; and faecal calprotectin (FC), a marker which is elevated in patients with intestinal inflammation [Summerton et al., 2002; Costa et al., 2003]. These were compared to urinary VOCs, measured using a Lonestar FAIMS device (Owlstone, UK) as described in Section 2.3.

Both stool markers have been suggested as potential rule-out tests for colorectal cancer [Mowat et al., 2016; Røseth et al., 1993; Tibble et al., 2001], and both are recommended for clinical use by the UK's NICE. Quantitative FIT testing is recommended by NICE to guide referral in primary care for suspected colorectal cancer in patients without rectal bleeding [NICE, 2017], using elevated F-Hb as a rule-in test to the TWW urgent care pathway for symptomatic patients whose symptoms alone fail to reach the >3% PPV threshold for urgent referral. FC is recommended by

NICE for differential diagnosis between inflammatory and non-inflammatory bowel diseases, provided cancer is not suspected [NICE, 2013].

The patient cohort

Patients were recruited from referrals to the TWW urgent lower gastrointestinal pathway at UHCW between January 2015 and September 2016. 1,850 patients were approached for inclusion in the study. 834 patients were excluded due to non-attendance, cancelling their appointments, declining to participate, language barriers, visual impairment, illness which rendered them unable to provide valid consent, or being deemed physically unfit for further investigation by the clinician. The remaining 1,016 patients were recruited prospectively into the study. All recruited patients underwent complete colonic investigations. 562 patients returned both stool and urine samples and were included in the statistical analysis.

Metrics

As a screening test it is necessary to minimise the number of false negatives (people who are incorrectly screened out of further investigation). However, a trivial solution to minimizing this is to return a positive result for every patient, which clearly provides no benefit. In this study we set a minimum acceptable NPV of 98.5% (in line with previous published studies [Widlak et al., 2017]), then chose the threshold which maximized the specificity of the test.

Combining F-Hb and FC

In clinical practice in the UK F-Hb and FC are thresholded to determine diagnostic outcomes. A threshold of $10 \mu\text{g g}^{-1}$ is used with F-Hb, with patients exceeding the threshold referred to secondary care for further investigation [NICE, 2017]. For FC the diagnostic guidance is more complex, but broadly speaking two or more tests within two weeks of $100\text{--}250 \mu\text{g g}^{-1}$ or one test of greater than $250 \mu\text{g g}^{-1}$ leads to a patient being referred to secondary care [NICE, 2013].

For this study we investigated methods for combining F-Hb and FC which go beyond thresholding both biomarkers. Specifically, we examined logistic regression, logistic regression with interactions, robust logistic regression, independent robust mixture models (IRMM), independent robust mixture models with implicit subclasses and censoring (IRMM-IS-C), and Random Forests. With the exception of Random Forests we implemented these models in JAGS, “a program for analysis

of Bayesian graphical models using Gibbs sampling” [Plummer et al., 2003], using the rjags R package to call JAGS from R [Plummer, 2016]. For details of the JAGS implementations see Appendix A.

Censoring of stool biomarkers

The measurements for both F-Hb and FC were censored by the laboratory. F-Hb was right-censored, with results above $1000 \mu\text{g g}^{-1}$ reported as $1000 \mu\text{g g}^{-1}$. FC was left-censored, with results below $15 \mu\text{g g}^{-1}$ reported as $15 \mu\text{g g}^{-1}$. The effect of censoring on prediction performance was investigated by the inclusion of the IRMM-IS-C model, which accounts for the censored data.

VOC analysis methods

For this study, the statistical analysis of the VOCs drew on techniques established in previously published studies using FAIMS-derived VOC data [Covington et al., 2013; Arasaradnam et al., 2014; Westenbrink et al., 2015; Covington et al., 2015]. The raw FAIMS data were first wavelet-transformed,¹⁰ and the highest frequency components discarded. Feature selection was then performed using a MWW test [Mann and Whitney, 1947] at a significance level of 0.8, with multi-test correction performed using sequential goodness-of-fit [Carvajal-Rodríguez et al., 2009]. This yielded a set of $\sim 10^4$ variables. The dimensionality of the selected variables was then reduced using PCA, with the number of principal components to retain selected automatically as described in Minka [2001]. A logistic regression model was then trained on the resulting set of features. Predictions were generated using 10-fold cross-validation [Kohavi et al., 1995], with both feature selection and model training occurring inside the cross-validation process. Wavelet transformation, as an unsupervised transformation, was performed prior to cross-validation to reduce computation time.

Results

F-Hb and FC Figures 2.10 and 2.11 show the ROC curves for using thresholding of F-Hb and FC respectively to diagnose colorectal cancer, high risk adenomas (those over 10mm in size), and all adenomas (including high risk adenomas). F-Hb significantly outperforms FC for all three diseases, and FC performed no better than chance for all adenomas and high risk adenomas. A threshold of $97.6 \mu\text{g g}^{-1}$ for F-Hb maximised specificity under the constraint that NPV was greater than 0.985 (details in Table 2.1). No threshold for FC achieved an NPV greater than 0.985.

¹⁰Using Daubechies D2 extremal-phase wavelets with symmetric boundary conditions.

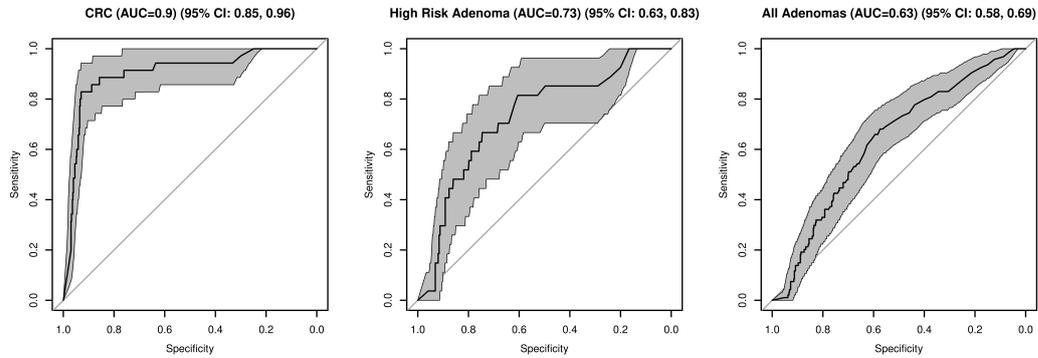


Figure 2.10: ROC curves for the prediction of colorectal cancer (CRC), high risk adenomas (>10mm in diameter), and all adenomas using thresholding of F-Hb. 95% confidence intervals are shown in grey.

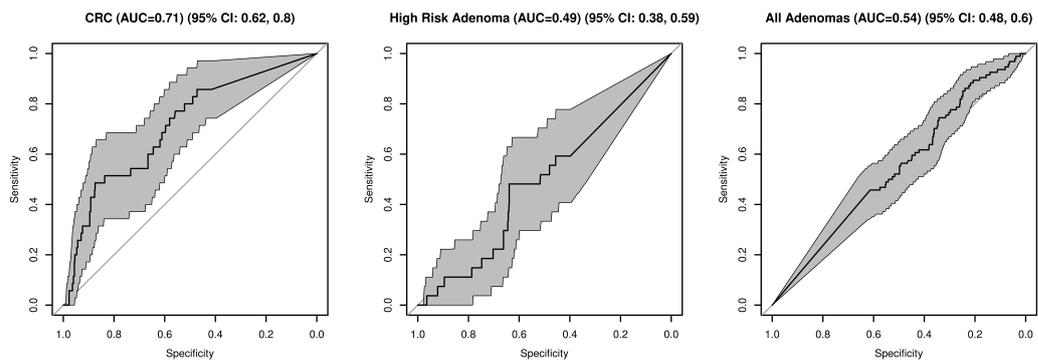


Figure 2.11: ROC curves for the prediction of colorectal cancer (CRC), high risk adenomas (>10mm in diameter), and all adenomas using thresholding of FC. 95% confidence intervals are shown in grey.

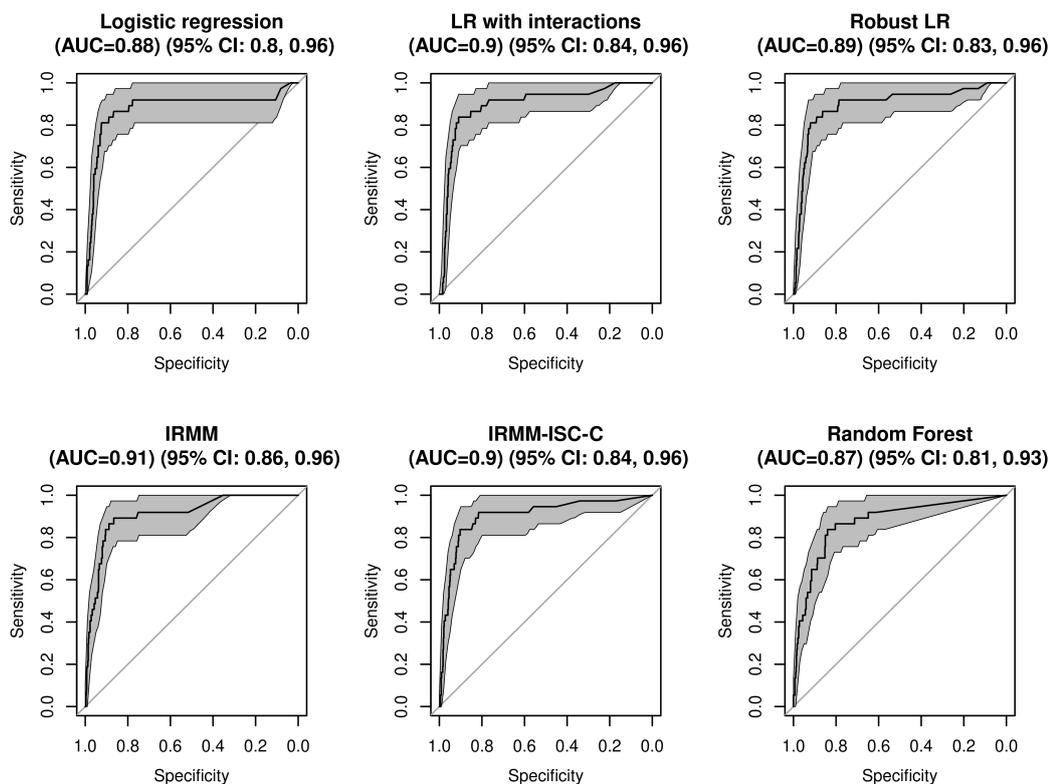


Figure 2.12: ROC curves for the six models investigated for combining F-Hb and FC to predict colorectal cancer. 95% confidence intervals are shown in grey. No model significantly outperformed F-Hb alone, the ROC curves for which are shown in Figure 2.10.

Combining F-Hb and FC Table 2.1 gives the performance of the various models for combining F-Hb and FC alongside the performance of F-Hb alone, and Figure 2.12 shows the ROC curves for the combined biomarker models. No model performed significantly better than F-Hb. Of particular note is the fact that IRMM-ISC-C, which models the censoring in the biomarker data, performs no better than the models which do not account for censoring. Intuitively this is to be expected since the censoring cut-offs ($<15 \mu\text{g g}^{-1}$ for FC and $>1000 \mu\text{g g}^{-1}$ for F-Hb) only censor a small number of extreme results. Despite this, this is an encouraging result in the wider context, since laboratories often apply censoring to values which are either below the detection threshold of the test (as in FC), or extreme values outside of the normal range (as in F-Hb). Figure 2.13 shows the predicted probabilities of cancer for each model, broken down by the true disease status of the patient.

	Model	NPV	Sensitivity	Specificity	AUC
	F-Hb	0.99 (0.97, 0.99)	0.78 (0.64, 0.91)	0.94 (0.91, 0.95)	0.90 (0.85, 0.96)
Logistic regression (LR)		0.99 (0.97, 0.99)	0.78 (0.64, 0.91)	0.93 (0.90, 0.95)	0.88 (0.80, 0.96)
	LR with interactions	0.99 (0.97, 0.99)	0.78 (0.64, 0.91)	0.93 (0.90, 0.95)	0.90 (0.84, 0.96)
	Robust LR	0.99 (0.97, 0.99)	0.78 (0.64, 0.91)	0.93 (0.91, 0.95)	0.89 (0.83, 0.96)
	IRMM	0.99 (0.97, 0.99)	0.78 (0.65, 0.91)	0.92 (0.89, 0.94)	0.91 (0.86, 0.96)
	IRMM-ISC-C	0.99 (0.98, 1.0)	0.81 (0.68, 0.93)	0.91 (0.89, 0.93)	0.90 (0.84, 0.96)
	Random Forest	0.99 (0.97, 1.0)	0.81 (0.68, 0.93)	0.85 (0.82, 0.88)	0.87 (0.81, 0.93)

Table 2.1: Table of NPV, sensitivity, and specificity for the different models for combining F-Hb and FC results, as well as for raw F-Hb results, where the threshold for a positive result has been picked to maximise specificity under the constraint that NPV is greater than 0.985. Brackets show the 95% confidence interval. No model significantly outperformed F-Hb, suggesting that FC does not provide any additional information about colorectal cancer status that is not provided by F-Hb. Note that the specificity of the Random Forest model is significantly lower (at the 5% level) than that of any of the Bayesian models examined.

Augmenting stool biomarkers with clinical data Table 2.2 shows the result of including clinical data (age, gender, ethnicity, medical history) in a subset of the models considered, and Figure 2.14 shows the associated ROC curves. As before, no significant improvement is seen over F-Hb alone.

	Model	NPV	Sensitivity	Specificity	AUC
	F-Hb	0.99 (0.97, 0.99)	0.78 (0.64, 0.91)	0.94 (0.91, 0.95)	0.90 (0.85, 0.96)
Logistic regression		0.99 (0.98, 1)	0.81 (0.68, 0.93)	0.9 (0.88, 0.92)	0.92 (0.88, 0.96)
	IRMM	0.99 (0.98, 1)	0.81 (0.68, 0.93)	0.91 (0.88, 0.93)	0.89 (0.82, 0.96)
	Random Forest	0.99 (0.97, 0.99)	0.78 (0.64, 0.91)	0.92 (0.9, 0.94)	0.90 (0.83, 0.96)

Table 2.2: Table of NPV, sensitivity, and specificity for the different models for combining F-Hb and FC results with additional clinical information, as well as for raw F-Hb results without any clinical information for comparison. The clinical data included were age, gender, ethnicity, and medical history. As before the threshold for a positive result has been picked to maximise specificity under the constraint that NPV is greater than 0.985. Brackets show the 95% confidence interval. No model significantly outperformed F-Hb alone, suggesting that the inclusion of clinical data does not provide any additional information about colorectal cancer status that is not provided by F-Hb.

VOCs Figure 2.15 shows the ROC curves for using VOCs to diagnose colorectal cancer, high risk adenomas, and adenomas. For colorectal cancer VOCs achieve an AUC of 0.67 (95% CI:0.57, 0.77), while for adenomas and high risk adenomas they do

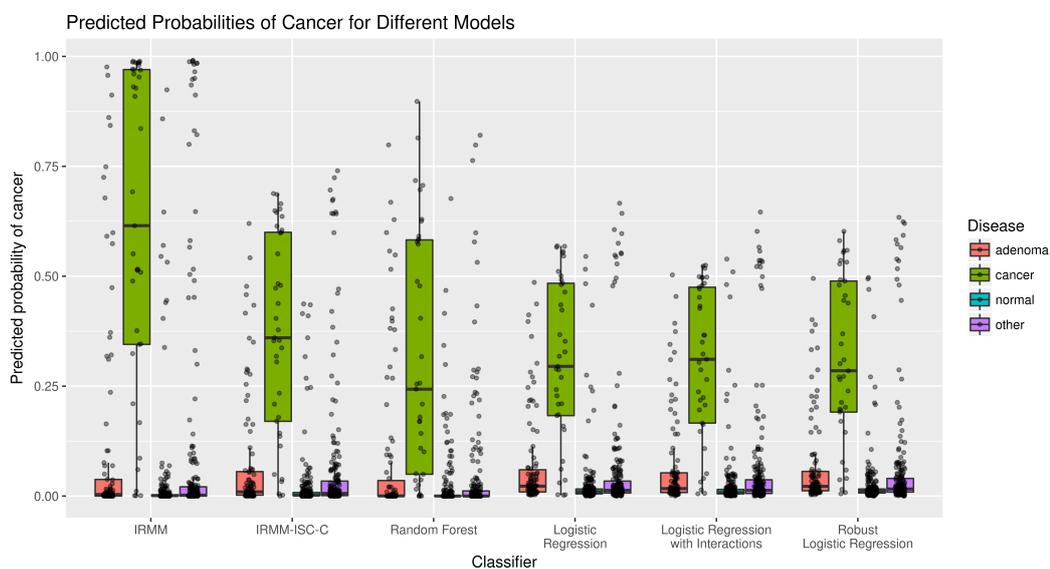


Figure 2.13: Predicted probability of cancer for the different methods of combining F-Hb and FC, broken down by true disease status. Probabilities were generated using a ten-fold cross-validation.

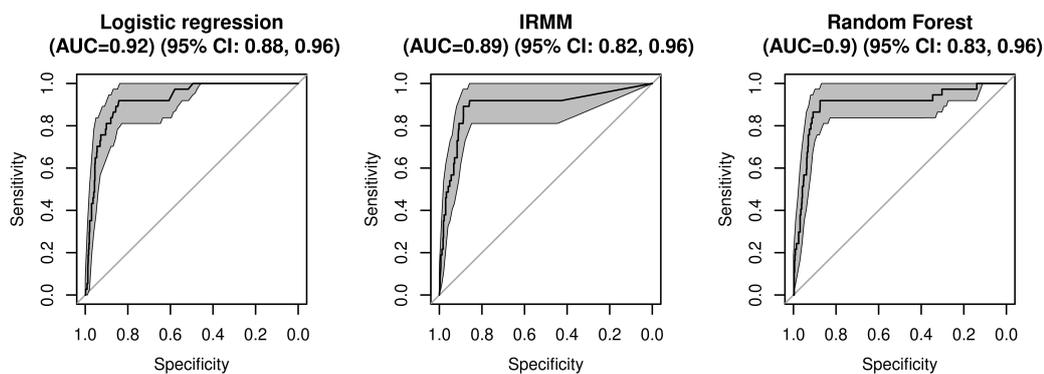


Figure 2.14: ROC curves for the three models investigated for combining F-Hb, FC and clinical data to predict colorectal cancer. 95% confidence intervals are shown in grey. No model significantly outperformed F-Hb alone, the ROC curves for which are shown in Figure 2.10.

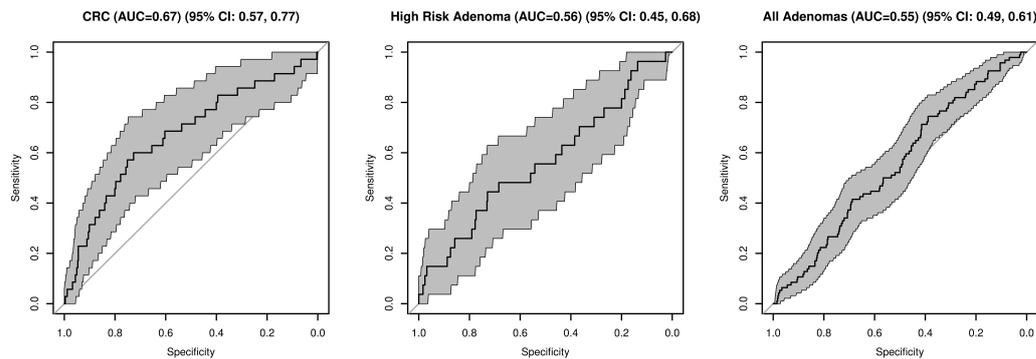


Figure 2.15: ROC curves for the prediction of colorectal cancer (CRC), high risk adenomas (>10mm in diameter), and all adenomas using VOCs. VOCs perform no better than chance for the diagnosis of adenomas and high risk adenomas.

not perform significantly better than chance with AUCs (95% confidence intervals) of 0.55 (0.49, 0.61) and 0.56 (0.45, 0.68) respectively. No threshold gives an NPV greater than 0.985, however at a relaxed minimum NPV of 0.965 it is possible to obtain an NPV of 0.97 (0.95, 0.98) with sensitivity and specificity of 0.69 (0.52, 0.83) and 0.60 (0.56, 0.65) respectively. With the same relaxed NPV constraint, F-Hb attains an NPV of 0.97 (0.95, 0.98), with a sensitivity and specificity of 0.49 (0.32, 0.65) and 0.96 (0.95, 0.98) respectively.

Compliance rates for different sample types

In part this study was motivated by the poor uptake rates for faecal tests such as FIT and FC. For faecal tests where a patient is provided with a kit for taking a stool sample at home, around half of patients do not return a sample for testing, with reported compliance rates of between 14 and 61% [Adler et al., 2014; Palmer et al., 2014; Stokamer et al., 2005]. In contrast, at UHCW where the study was conducted urine tests have a compliance rate of 86%, and stool tests have a compliance rate of 67% [R Arasaradnam 13th July 2018, personal communication].

It is plausible that patient compliance is influenced by the symptoms they are suffering from, and therefore by their underlying disease (or lack thereof). Assuming the test result depends on the disease state of the patient, this would imply that compliance and test outcome are not independent, making it impossible to adjust test metrics to account for compliance without some attempt to quantify the difference

in test results between compliant and non-compliant patients.¹¹ However, there is evidence that this is not the case. Stokamer et al. [2005] investigated the effect of intensive one-on-one patient education on FOB test compliance, and found that the intervention group were more likely to comply with FOB testing than the control group, with uptake rates of 65.9% and 51.3% respectively. Of particular interest is the fact that the proportion of patients who had a positive FOB test did not differ between the two groups (4.6% vs 6.0%, $p=0.51$), suggesting that compliance is not significantly influenced by test outcome.

Given this assumption, it is possible to adjust the test metrics sensitivity, specificity, PPV, and NPV to account for compliance rates, where (in the context of a screening test) non-compliance is treated as a negative test result, since in the absence of a result no follow-up will occur. In this case, the sensitivity and NPV decrease, the specificity increases, and the PPV is unchanged, as shown below.

First we calculate adjusted values for the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). α is the compliance rate.

$$TP_{\text{adj}} = \alpha \cdot TP \quad (2.3)$$

$$FP_{\text{adj}} = \alpha \cdot FP \quad (2.4)$$

$$\begin{aligned} TN_{\text{adj}} &= TN + (1 - \alpha) \cdot FP \\ &= TN + (1 - \alpha) \cdot (N - TN) \\ &= \alpha \cdot TN + (1 - \alpha) \cdot N \end{aligned} \quad (2.5)$$

$$FN_{\text{adj}} = (1 - \alpha) \cdot TP + FN \quad (2.6)$$

Sensitivity, specificity, PPV, and NPV can then be adjusted as follows:

$$\text{sens}_{\text{adj}} = \frac{TP_{\text{adj}}}{P} \quad (2.7)$$

$$= \alpha \cdot \text{sens}$$

$$\begin{aligned} \text{spec}_{\text{adj}} &= \frac{TN_{\text{adj}}}{N} \\ &= \frac{\alpha \cdot TN + (1 - \alpha) \cdot N}{N} \end{aligned} \quad (2.8)$$

$$= 1 - \alpha(1 - \text{spec})$$

¹¹Which, given the non-compliance of one group, would seem to be a very difficult task indeed.

$$\begin{aligned}
\text{PPV}_{\text{adj}} &= \frac{\text{TP}_{\text{adj}}}{\text{TP}_{\text{adj}} + \text{FP}_{\text{adj}}} \\
&= \frac{\alpha \cdot \text{TP}}{\alpha \cdot \text{TP} + \alpha \cdot \text{FP}} \\
&= \frac{\text{TP}}{\text{TP} + \text{FP}} \\
&= \text{PPV}
\end{aligned} \tag{2.9}$$

$$\begin{aligned}
\text{NPV}_{\text{adj}} &= \frac{\text{TN}_{\text{adj}}}{\text{TN}_{\text{adj}} + \text{FN}_{\text{adj}}} \\
&= \frac{\alpha \cdot \text{TN} + (1 - \alpha) \cdot \text{N}}{\alpha \cdot \text{TN} + (1 - \alpha) \cdot \text{N} + (1 - \alpha) \cdot \text{TP} + \text{FN}} \\
&= \frac{\alpha \cdot \text{TN} + (1 - \alpha) \cdot \text{N}}{\alpha \cdot \text{TN} + (1 - \alpha) \cdot (\text{N} + \text{TP}) + \text{FN}} \\
&= \frac{\alpha \cdot \text{TN} + (1 - \alpha) \cdot \text{N}}{\alpha \cdot (\text{TN} + \text{FN}) + (1 - \alpha) \cdot (\text{N} + \text{TP} + \text{FN})} \\
&= \frac{\alpha \cdot \text{TN} + (1 - \alpha) \cdot \text{N}}{\alpha \cdot (\text{TN} + \text{FN}) + (1 - \alpha) \cdot (\text{N} + \text{P})}
\end{aligned} \tag{2.10}$$

If adjusting test metrics as above, an adjustment is also necessary to the way in which confidence intervals are calculated. If using a bootstrap estimate of the CI, it is insufficient to simply perform a bootstrap using the adjusted statistics, since this will underestimate the variance introduced by the compliance/noncompliance process. Instead, model compliance in the bootstrap sample by independently setting each result in the bootstrap sample to be negative with a probability of α , the compliance rate.

More formally, let x be the vector of predicted results for a bootstrap sample of size n , with $x_i = 0$ for a negative prediction and $x_i = 1$ for a positive prediction. Now, draw c_i from a Bernoulli distribution with probability α for $i \in \{1, \dots, n\}$. $c_i = 0$ represents non-compliance for sample i . Then, set $\tilde{x}_i = 0$ when $c_i = 0$ and $\tilde{x}_i = x_i$ otherwise, and calculate test metrics as normal (without adjusting for compliance analytically) using \tilde{x} in place of x . Repeat this procedure for each bootstrap sample until sufficient bootstrapped statistics have been calculated to estimate the confidence interval. A comparison between the bootstrapped statistic distributions using the naïve method (using equations 2.7–2.10 directly as per a normal bootstrap) and the correct method described above is show in Figure 2.16.

Applying the results in equations 2.7–2.10 to the metrics in this study allows us to more accurately assess the real-world effectiveness of VOCs and F-Hb as screening

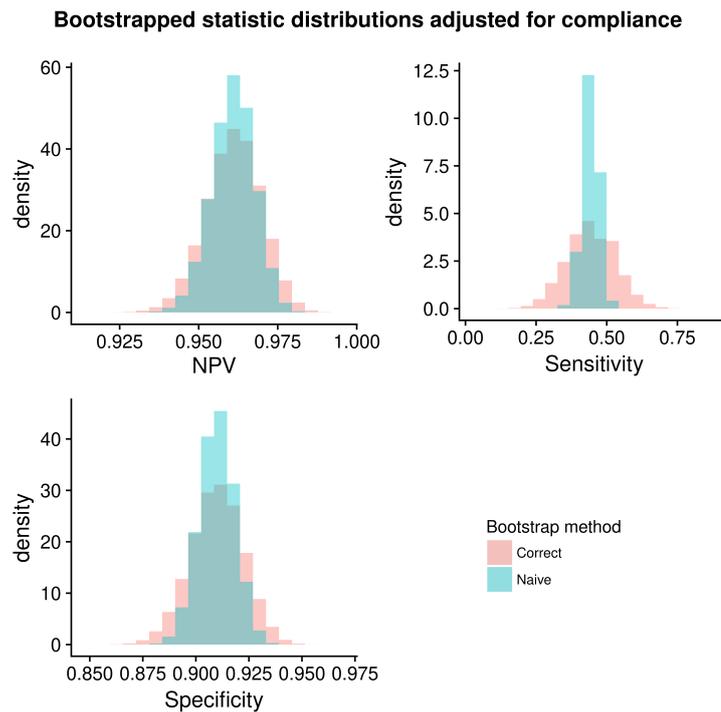


Figure 2.16: Example bootstrap distributions for the NPV, sensitivity, and specificity of F-Hb testing, adjusted to take into account the compliance rate of the test. The histograms show the statistics calculated naïvely using the equations in equations 2.7–2.10, and calculated correctly by modelling compliance in the bootstrapping process as described in subsection 2.4.1. Note that the naïve bootstrap method underestimates the variance of the statistics.

<i>Minimum NPV</i>	<i>Biomarker</i>	<i>NPV</i>	<i>Sensitivity</i>	<i>Specificity</i>
0.970	F-Hb	0.970 (0.956, 0.984)	0.543 (0.382, 0.703)	0.955 (0.936, 0.970)
0.960	F-Hb	0.960 (0.943, 0.976)	0.380 (0.225, 0.543)	0.967 (0.952, 0.981)
0.960	VOCs	0.960 (0.939, 0.979)	0.590 (0.422, 0.750)	0.659 (0.619, 0.698)

Table 2.3: Compliance-adjusted statistics for F-Hb and urinary VOCs for detecting colorectal cancer in the TWW study. 95% confidence intervals (calculated using the adjusted bootstrap process for compliance-adjusted statistics described in the main text) are shown in brackets.

tests for colorectal cancer, taking into account differing compliance rates. Using the compliance rates for stool and urine samples at UHCW, where the study was conducted, of 67% and 86% respectively [R Arasaradnam 13th July 2018, personal communication], we found that F-Hb no longer has a threshold at which it achieves an NPV of at least 0.985. Stepping down the threshold in increments of 0.005, the first viable threshold is at an NPV of 0.970, with a threshold of $89.1 \mu\text{g g}^{-1}$, NPV of 0.970 (0.956, 0.984), and sensitivity and specificity of 0.543 (0.382, 0.703) and 0.955 (0.936, 0.970) respectively.

For VOCs, the first viable NPV constraint is 0.960, with an NPV of 0.960 (0.939, 0.979), and sensitivity and specificity of 0.590 (0.422, 0.750) and 0.659 (0.619, 0.698) respectively. At this NPV constraint, F-Hb achieves an NPV of 0.960 (0.943, 0.976), and sensitivity and specificity of 0.380 (0.225, 0.543) and 0.967 (0.952, 0.981) respectively. These results are summarised in Table 2.3.

Discussion

At first sight, F-Hb significantly outperforms both FC and VOCs as a screen-out test for colorectal cancer in high-risk patients. Its AUC for detecting colorectal cancer is significantly better than either of the other tests considered, it is the only test to perform better than chance for the detection of adenomas, and it is the only test to achieve an NPV greater than 0.985.

The ability to achieve an acceptable NPV in particular is a key factor in clinical acceptance of a screen-out test, due to the high costs—to clinical outcome and reputation as well as financially—associated with incorrectly screening people out of an urgent care pathway

However, when compliance based on the test medium (stool vs urine) is taken into account, the difference between VOCs and F-Hb becomes less stark. The difference

in achievable NPV is reduced, with F-Hb no longer achieving the target minimum NPV of 0.985. However, F-Hb still outperforms VOCs, with a better NPV and specificity. Given the small adjusted difference between the two, it seems plausible that by combining FAIMS VOC data with tests of other sampling mediums with high compliance rates a test could be developed which outperforms F-Hb when compliance is accounted for.

As a final note, the fact that the addition of other clinical data (demographics and medical history) did not improve test performance could be explained by the fact that this data has already been used in primary care to refer patients to the TWW urgent care pathway, and as such patients whose clinical data indicates they are low risk for colorectal cancer would not be present in the study population.

2.4.2 Colorectal cancer screening using urinary VOCs

In collaboration with a team from UHCW, we investigated the use of urinary VOCs as a screening test for colorectal cancer and colorectal polyps. The study followed Arasaradnam et al. [2014], a prospective case control study discussed in section 2.1.3.

The patient cohort

All patients with a positive faecal occult blood test attending the bowel cancer screening nurse clinic at UHCW were eligible. Clinical outcomes for the study (colorectal cancer, adenoma, other colorectal disease, or normal colonoscopy) were determined by colonoscopy or CT colonography.

127 patients were recruited into the study, of whom 11 were diagnosed with colorectal cancer, 57 with adenomas, 32 with other colorectal diseases, and 27 with normal colonoscopies. 57% of patients were male, with a median age of 68.

Sample handling and FAIMS analysis

Demographic and clinical data were recorded and two 20ml urine samples were collected when the patient attended the clinic. Samples were stored at -80°C for batch analysis on a Lonestar FAIMS unit (Owlstone, UK), and were heated to 38°C by an ATLAS sampling system (Owlstone, UK) prior to sampling by the FAIMS unit. Four full scans of the compensation voltage and dispersion field strength were performed for each sample.

	<i>AUC (95% CI)</i>		
	<i>Cancer</i>	<i>Polyps</i>	<i>Cancer or Polyps</i>
Random Forest	0.48 (0.39, 0.57)	0.60 (0.50, 0.69)	0.72 (0.63, 0.81)
Logistic regression	0.58 (0.42, 0.74)	0.64 (0.54, 0.73)	0.68 (0.59, 0.78)
Sparse logistic regression	0.50 (0.30, 0.71)	0.67 (0.57, 0.76)	0.69 (0.60, 0.79)
Stochastic gradient boosting	0.57 (0.37, 0.77)	0.62 (0.52, 0.71)	0.70 (0.61, 0.80)
Linear SVM	0.57 (0.39, 0.75)	0.63 (0.53, 0.72)	0.68 (0.58, 0.77)
RBF SVM	0.56 (0.39, 0.74)	0.61 (0.51, 0.71)	0.69 (0.60, 0.79)

Table 2.4: AUC for a number of classifiers for the detection of colorectal disease using urinary VOCs. Results are shown for the detection of cancer, polyps, and either cancer or polyps. 95 % confidence intervals are shown in brackets. No classifier was successful at detecting cancer, however curiously performance improved when trained on a positive class of cancer and polyps over cancer or polyps alone. It is likely that the number of incidences of cancer (11) was too low to detect any effect.

Statistical analysis

Statistical analysis was performed using our `FAIMSToolkit` R package. Noise variables were identified and zeroed out using standard deviation thresholding, with 75 % of variables removed. The data was then wavelet transformed, with the highest frequency components discarded as noise. Components with variance were then removed, and the wavelet transformed data was scaled to have zero mean and unit variance. The top 1024 features were then selected using the MWW test, reduced in dimensionality via PCA, and passed to a number of classifier algorithms to assess their performance on the data. AUC, sensitivity and specificity were assessed using 10-fold cross-validation, with confidence intervals calculated using bias corrected and accelerated (BCa) bootstrap intervals [DiCiccio and Efron, 1996]. We assessed the ability of urinary VOCs to differentiate between: cancer versus no cancer; polyps versus no polyps; and cancer or polyps vs neither cancer nor polyps.

Results

Table 2.4 shows the area under the ROC curve results for a range of classifiers (detailed in section 2.3.6) for the conditions studied. No classifier performed better than chance for detection of cancer. For detection of polyps, all classifiers performed better than chance, albeit with mediocre performance. Interestingly, combined detection of polyps and cancers appears to give improved performance. For each of the three classification tasks all classifiers achieved comparable performance.

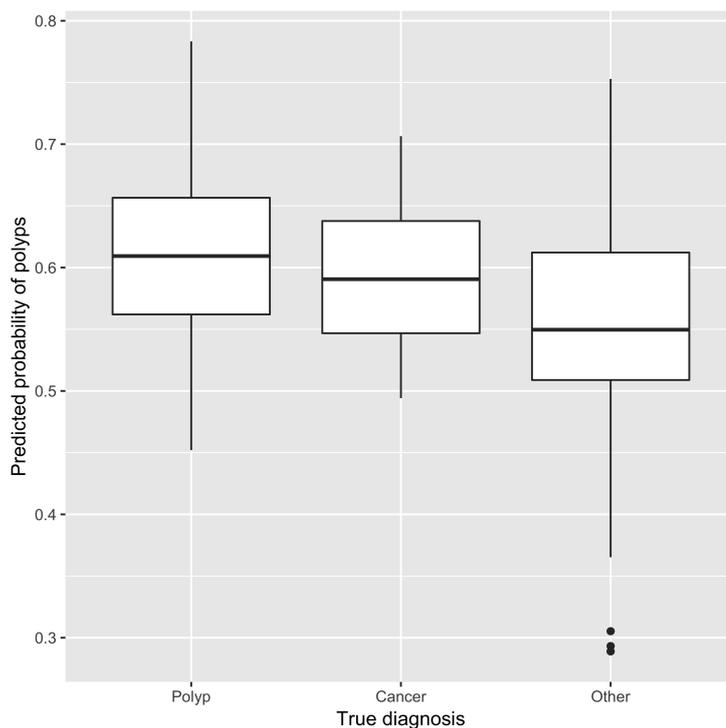


Figure 2.17: Predicted probability of polyps broken down by true disease status. The “Other” class includes both other colorectal disease and no disease. It can be seen that the distribution of predicted probabilities for the cancer patients is closer to that of polyps patients than others, leading to misclassification if they are classified separately using a binary classifier.

Discussion

Interestingly, the combination of cancers and polyps into a single class led to better performance than either condition approached singly. This suggests that cancers and polyps present similarly in the urinary VOC data, leading to misclassification and poor performance when classified separately. By way of example, Figure 2.17 shows the distribution of predicted probabilities of polyps by sparse logistic regression, broken down by true disease status. The distribution for cancer patients can be seen to be similar to that for polyp patients.

Unfortunately we were brought in to the study after the study design had been finalised, and as such were not able to contribute power calculations for the study in advance. This study suffered from the patient cohort containing an unusually low proportion of patients with colorectal cancer, which led to a small pool of

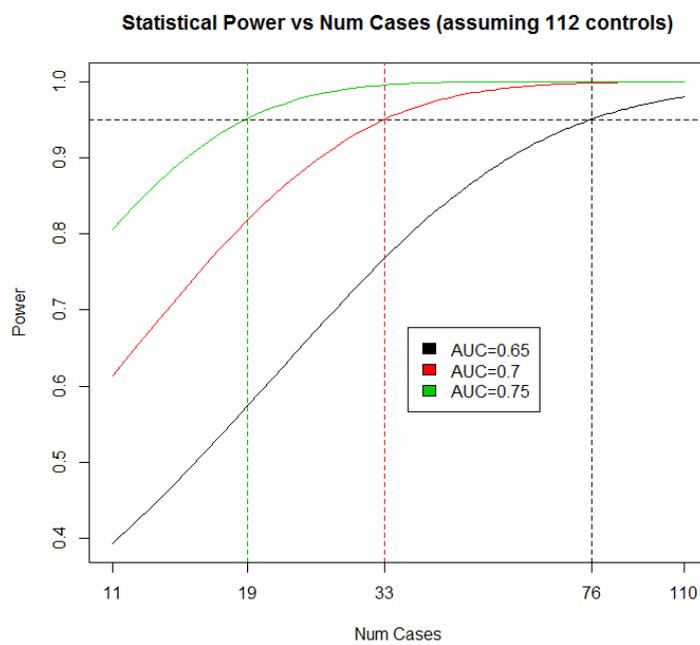


Figure 2.18: Power calculations for the colorectal cancer screening study. Three plausible effect sizes were assessed, with intercept lines showing the number of cases required for each effect size to achieve 95% statistical power.

Effect size (AUC)	Cases required for $n\%$ power	
	95	99
0.65	76	> 110
0.70	33	52
0.75	19	29

Table 2.5: Power calculations for the colorectal cancer screening study. 99 % power at an effect size of 0.65 required the recruitment of more cases than controls.

positive training examples and significantly reduced the statistical power of the study. With this in mind, after the study had concluded we reviewed the possibility of enriching the patient cohort with additional colorectal cancer patients. To this end we conducted power calculations to assess how many additional patients with colorectal cancer would be need to be recruited into the study to improve its statistical power, assuming the number of control cases remained the same. The results of this analysis can be seen in Figure 2.18 and Table 2.5. Assuming a true AUC of 0.7 (based on the AUC observed for detecting polyps and cancers together), an addition 22 cancer patients would need to be recruited for the study to have a statistical power of 95 %.

2.4.3 Distinguishing colorectal cancer from IBS

This study was intended to investigate whether urinary VOCs could be used to distinguish between colorectal cancer and IBS. Unfortunately a number of problems led to the results of the study being rendered useless. However, this study highlights two issues which can arise and must be guarded against when working with FAIMS data: sensor drift, and failing to randomize sample analysis order. During the study the FAIMS instrument suffered from an initially undetected failure of its sensor chip, with the sensor degrading over the course of several months. This was ultimately diagnosed through visual assessment of the FAIMS output by a member of the lab who was experienced with the FAIMS. The data were determined to be unusable due to the instrument failure. However, initially the instrument failure was masked by a problem with the study design which led to excellent classifier performance. The AUC found in the study so far exceeded that of previous studies that we investigated possible confounding factors with our collaborators. We finally determined that the samples had not been properly randomized when analysed by the FAIMS: the urine samples for colorectal cancer patients had been analysed by one person, while the samples for IBS patients had been analysed by a different person.

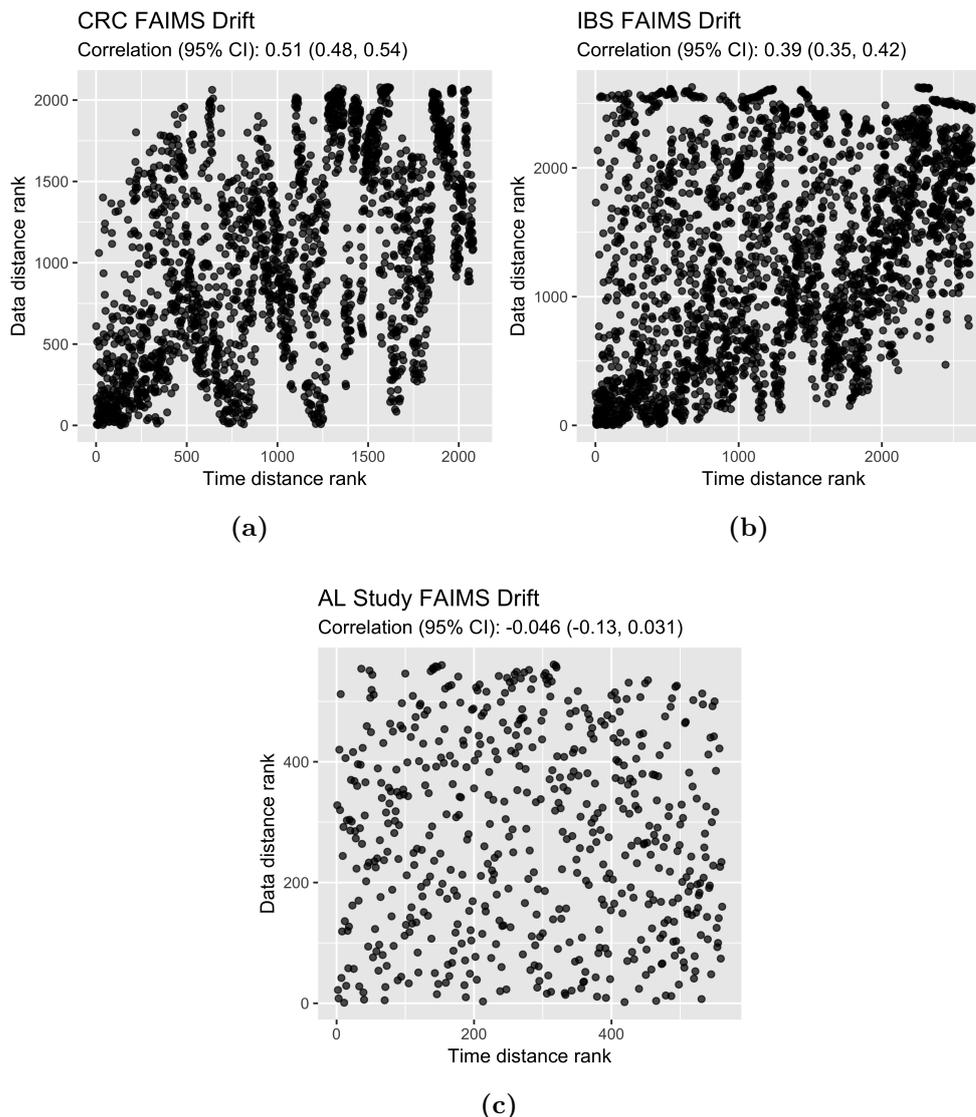


Figure 2.19: Plots for detecting FAIMS drift. For each pair of samples we calculated the L_2 distance between their FAIMS data matrices, and the time in seconds between the times at which they were analysed on the FAIMS device. We then plotted each sample-pair's time distance rank against its data distance rank. If the two are uncorrelated, one would expect the points to be uniformly distributed over the $[0, n]^2$ square, where n is the number of samples. If the sensor is drifting, one would expect samples which are run further apart to differ more than samples which were run closer together. (a) and (b) show the plots for the colorectal cancer and IBS samples respectively from the CRC vs IBS study in section 2.4.3. (c) shows the same plot for the AL study discussed in section 2.4.5. Drift is evident in (a) and (b), but not in (c).

The fact that the sensor failure had gone undetected led to the development of a simple test for sensor drift and failure. In the absence of sensor drift, or failure to randomize sample batch analysis properly, it is expected that the L_2 distance between two samples' data and the distance in time between the two samples will be uncorrelated. To detect this visually, for each pair of samples we calculated the L_2 distance between their FAIMS data matrices, and the time in seconds between the times at which they were analysed on the FAIMS device. We then plotted the sample-pair's time distance rank against its data distance rank. If the two are uncorrelated, one would expect a uniform random distribution on the $[0, n]^2$ square, where n is the number of samples. If the sensor is drifting, one would expect samples which are run further apart to differ more than samples which were run closer together. The latter is exactly what was seen when we produced this plot for this study, which can be seen in Figure 2.19. For comparison, the figure also shows the same plot for the study discussed in section 2.4.5, which shows the expected behaviour for a working FAIMS device.

2.4.4 Pre-symptomatic detection of anastomotic leakage following gastrointestinal surgery

In the context of early cancer detection, a drawback of the studies presented in this chapter are that the patients considered were already exhibiting symptoms which indicated an elevated risk of cancer. Due to cancer's low incidence¹² and the fact that many of the population of interest for pre-symptomatic screening are not presenting clinically, the cost of prospectively recruiting enough patients for a sufficiently powerful study of pre-symptomatic cancer detection is prohibitively expensive. The studies presented in the following two sections are therefore included to demonstrate the ability of VOC testing by FAIMS to detect disease early, before the presentation of clinical symptoms.

Anastomotic leakage

Anastomotic leakage (AL) is a common¹³ postoperative complication of colorectal surgery which can have lethal consequences. AL involves the breakdown of an

¹²For example, colorectal cancer—the focus of the other studies in this chapter—was the third most common malignant cancer in England in 2016 with age-standardised rates of 84.4 cases per 100,000 males and 55.4 cases per 100,000 females [Office for National Statistics, 2016].

¹³Reported leak rates vary depending on the site of the anastomosis, from a low of 1% for the stomach and small intestines to a high of 41% for the rectum [Turrentine et al., 2015].

anastomosis,¹⁴ and is associated with significant increases in morbidity, 30-day and long-term mortality, and treatment cost [Turrentine et al., 2015]. In collaboration with colleagues at VU University Medical Center, Amsterdam, we investigated whether urinary VOCs could be used to predict the development of AL in the first few days following major gastrointestinal surgery.

The patient cohort

We investigated VOCs as a biomarker for AL in patients who had undergone esophagectomy or pancreaticoduodenectomy to treat tumours of the esophagus or pancreatic head. 63 patients who were due to undergo esophagectomy ($n = 31$) or pancreaticoduodenectomy ($n = 32$) were recruited prospectively from the department of surgery at the VUmc (Amsterdam, The Netherlands). All study participants received standard treatment according to local protocol both pre- and post-operatively. Clinical postoperative AL was the primary endpoint of the study, as defined by the Esophagectomy Complications Consensus Group (ECCG) and International Study Group on Pancreatic Fistula (ISGPF) for esophagectomy and pancreaticoduodenectomy respectively. The presence of AL was confirmed by radiologic or endoscopic imaging following the presentation of clinical signs. Full details can be found in the published paper [Plat et al., 2018].

Sample handling and FAIMS analysis

For each patient, a urine sample was obtained each day for the three days immediately following surgery. Samples were frozen at -80°C within two hours of collection for batch processing by FAIMS. Samples were analysed using a Lonestar FAIMS unit (Owlstone, UK), and were heated to 38°C by an ATLAS sampling system (Owlstone, UK) prior to sampling by the FAIMS unit. Three full scans of the compensation voltage and dispersion field strength were performed for each sample. Figure 2.20 shows examples of the FAIMS data generated for an AL and non-AL patient. Full details of the FAIMS analysis process can be found in the published paper [Plat et al., 2018].

Statistical analysis

For this pilot study, the data for the three days were pooled to create datasets for each disease with enough data for cross-validation. As with the other studies in

¹⁴A surgically-introduced connection between two fluid-carrying body structures, for example the connection between two portions of bowel in a small bowel resection.

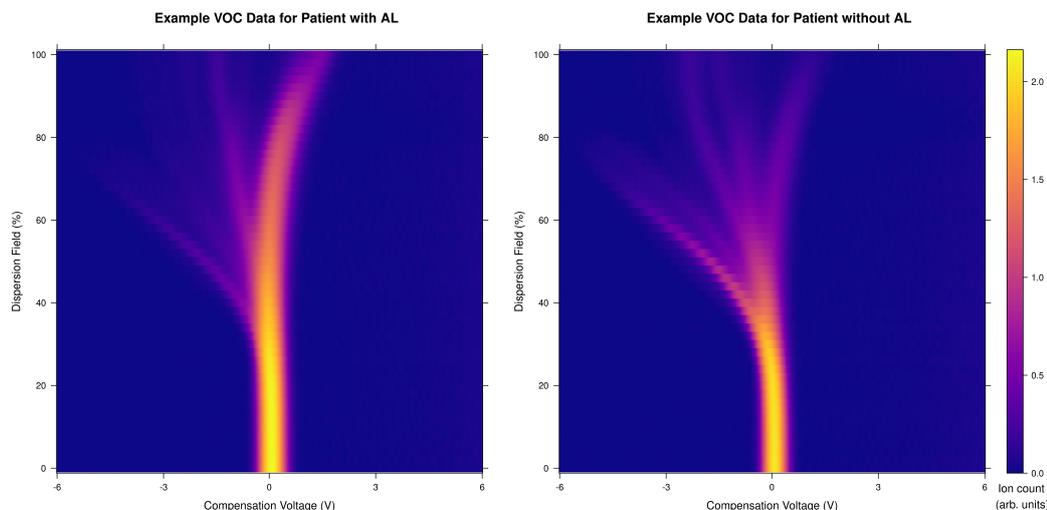


Figure 2.20: Example urinary VOC data for (on the left) a patient with AL, and (on the right) without AL, following pancreaticoduodenectomy. The samples shown were chosen to be the nearest to the mean for their respective classes (AL and non-AL) by L2-norm. The plots show the negative ion count for the final FAIMS run.

this chapter, the raw FAIMS data was wavelet-transformed using Daubechies D4 wavelets. Features were selected within cross-validation using a Wilcoxon rank-sum test, with the top scoring n features undergoing PCA, with the number of principal components to retain determined automatically for each fold using the technique described in Minka [2001]. To perform model selection an initial cross-validation was performed using pooled data from both diseases. The models included in the selection process were chosen to cover a wide range of model types: Random Forests [Breiman, 2001], boosting (stochastic gradient boosting) [Friedman, 2002], kernel methods (radial and linear SVMs) [Cortes and Vapnik, 1995], and sparse logistic regression [Friedman et al., 2010]. After model selection, the data was split by disease and a second cross-validation was performed with newly-generated folds.

Results

Of the 31 esophagectomy patients, 9 (29%) developed AL, with a median time to clinical presentation of 8 days (interquartile range (IQR) 18 days). 89 urine samples were collected, with one missing sample from the non-AL group and three missing samples from the AL group. FAIMS analysis failed for one non-AL sample, resulting in 64 non-AL and 24 AL samples being used in the final analysis.

<i>Model</i>	<i># features retained</i>	<i>AUC (95% CI)</i>
Random Forest	16	0.60* [†] (0.51, 0.69)
Sparse logistic regression	16	0.64* (0.55, 0.73)
SVM (radial kernel)	16	0.49 [†] (0.39, 0.59)
SVM (linear kernel)	16	0.50 [†] (0.39, 0.60)
Stochastic gradient boosting	16	0.58 [†] (0.49, 0.67)
Random Forest	64	0.71* (0.63, 0.80)
Sparse logistic regression	64	0.68* (0.60, 0.77)
SVM (radial kernel)	64	0.53 [†] (0.42, 0.63)
SVM (linear kernel)	64	0.53 [†] (0.43, 0.63)
Stochastic gradient boosting	64	0.64* [†] (0.55, 0.74)
Random Forest	128	0.69* (0.61, 0.78)
Sparse logistic regression	128	0.68* (0.60, 0.77)
SVM (radial kernel)	128	0.64* (0.54, 0.74)
SVM (linear kernel)	128	0.56 [†] (0.46, 0.65)
Stochastic gradient boosting	128	0.69* (0.61, 0.77)

Table 2.6: Model selection results for the anastomotic leakage pilot study. Random Forests, sparse logistic regression and stochastic gradient boosting performed well for 64 and 128 features retained, while both support vector machine (SVM) models performed poorly across the board. A Random Forest model retaining the top 64 features was selected for the study.

Statistically significant results (at a 5% level) are marked with an asterisk (*). Results which have a statistically significant (at the 5% level) difference in AUC to the selected model are marked with a dagger (†).

Of the 32 pancreaticoduodenectomy patients, 6 (19%) developed AL, with a median time to clinical presentation of 5 days (IQR 14 days). 95 urine samples were collected, with one missing sample from the AL group. FAIMS analysis failed for one non-AL sample, resulting in 77 non-AL and 17 AL samples being used in the final analysis.

The results of the model selection process are shown in Table 2.6. Random Forests, sparse logistic regression and stochastic gradient boosting performed similarly, with Random Forests outperforming other models with 64 or 128 features retained. The SVM models performed poorly across the board, and the linear kernel failed to perform better than chance for any of the feature subsets considered. Random Forests with 64 features retained was selected as the model for the study.

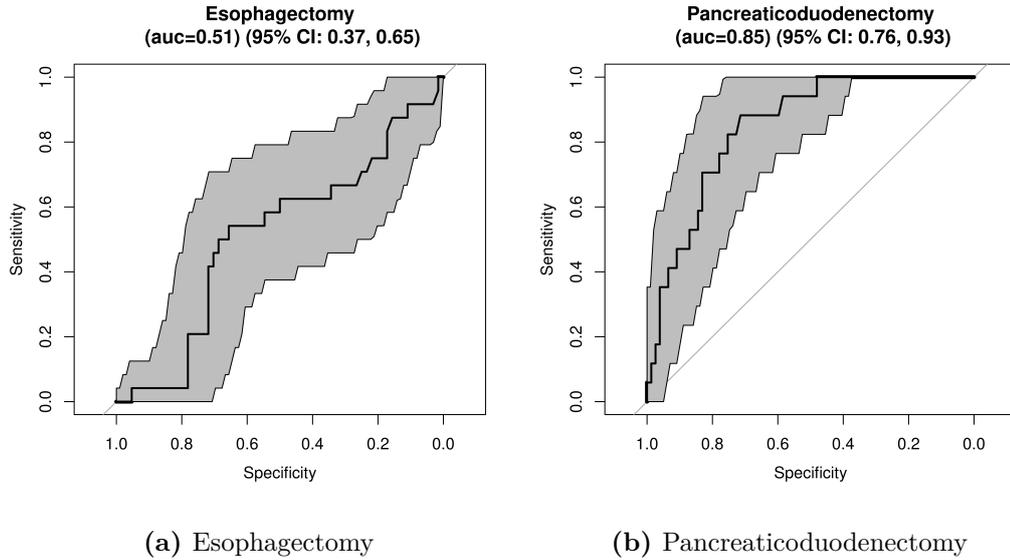


Figure 2.21: ROC curves for detecting anastomotic leakage after surgery using urinary VOCs and Random Forests.

<i>Surgery</i>	<i>AUC</i>	<i>Sensitivity</i>	<i>Specificity</i>
<i>Pancreaticoduodenectomy</i>	0.85 (0.76, 0.93)	0.76 (0.50, 0.93)	0.77 (0.66, 0.86)
<i>Esophagectomy</i>	0.51 (0.37, 0.65)	0.54 (0.33, 0.74)	0.55 (0.42, 0.67)

Table 2.7: Results for training a Random Forest classifier on urinary VOC data to detect anastomotic leakage after surgery. 95% confidence intervals are shown in brackets.

Figure 2.21 show the resulting ROC curves for detection of AL after esophagectomy and pancreaticoduodenectomy respectively. The classifier performed no better than chance for the detection of AL after esophagectomy with an AUC of 0.51 (95% CI: 0.37, 0.65), but performed well for the detection of AL after pancreaticoduodenectomy with an AUC of 0.85 (95% CI: 0.76, 0.93). More details are shown in Table 2.7. Figure 2.22 shows the distribution of the predicted probability of AL broken down by true disease status and surgery type.

Discussion

In this study we found evidence that urinary VOCs could provide a tool for diagnosing AL after pancreaticoduodenectomy significantly in advance of the onset of clinical signs. While the median time for clinical signs to appear was five days, VOC analysis was predictive of AL in the first three days after surgery. Figure 2.23 shows a

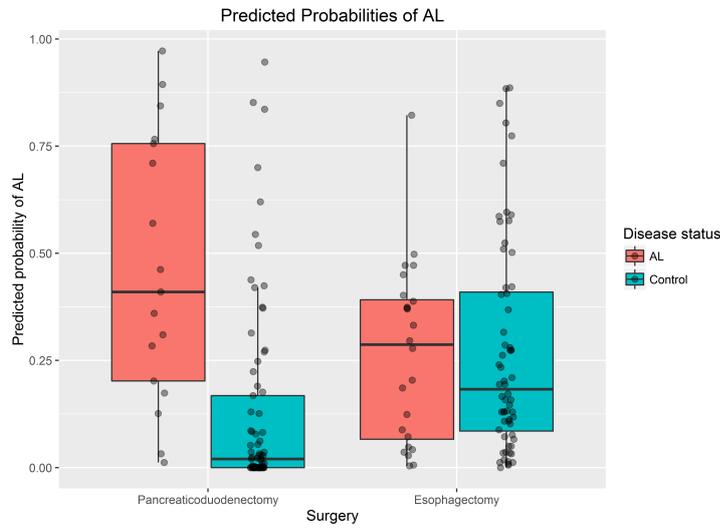


Figure 2.22: Predicted probability of anastomotic leakage (AL) broken down by surgery type. Pancreaticoduodenectomy has a good separation between AL and uncomplicated patients, while there is very poor separation for esophagectomy. Underlying values are plotted as points with a small jitter applied.

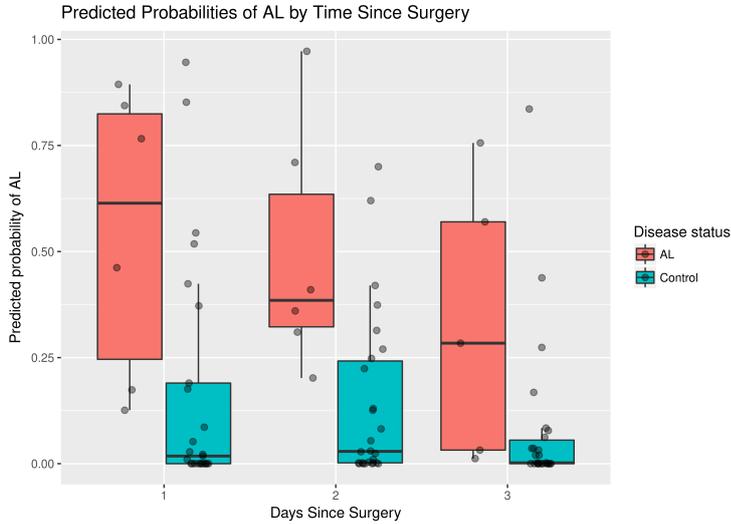


Figure 2.23: Predicted probability of anastomotic leakage (AL) for pancreaticoduodenectomy patients, broken down by the number of days after surgery that the urine sample was provided. VOC analysis demonstrates good sensitivity for detecting AL even one day post-surgery, before clinical signs are likely to manifest. Underlying values are plotted as points with a small jitter applied.

breakdown of the predicted probability of AL for pancreaticoduodenectomy patients' urine samples by true disease state and time since surgery. Even on the first day after surgery urinary VOCs show good separation between AL and non-AL samples, suggesting that urinary VOCs are able to detect the preclinical biochemical changes associated with the disease at a very early stage.

A significant problem with this study was the small sample size, which resulted in the need to pool multiple samples from the same patients into one dataset to provide enough data to train the classifier. The danger of this approach is that, since each patient contributed multiple urine samples, all of which are in the same class, it is possible that the classifier will learn to identify a particular patient's urine and use this to predict the AL-status of test data rather than learn patient-independent markers of AL. This could introduce a positive bias to the result of the cross-validation. However, the null result for detecting AL after esophagectomy provides a counter to this point - if the pooling had introduced a significant bias, a positive result would have been seen in both the esophagectomy and pancreaticoduodenectomy cohorts. The fact that a positive result was not observed for the esophagectomy cohort suggests that the size of the bias is not sufficient to invalidate the positive result observed for the pancreaticoduodenectomy cohort.

Finally, the result of the initial cross-validation is encouraging. The fact that a diverse range of models (Random Forests, stochastic gradient boosting, and sparse logistic regression) all produced similar AUCs suggests that the information present in the urinary VOCs is robust to the machine learning model used to make predictions, and is unlikely to be an artefact of the classification algorithm used.

2.4.5 Detection of anastomotic leakage following major colorectal surgery

Following the AL study discussed in Section 2.4.4, we were involved in a multi-centre¹⁵ cross-sectional cohort study investigating the ability of FAIMS analysis of urinary VOCs to detect AL following major colorectal surgery.¹⁶

¹⁵Five hospitals participated in the study: VU University Medical Center Amsterdam, West Fries Gasthuis Hoorn, Amphia Hospital Breda, Rode Kruis Hospital Beverwijk, and Spaarne Gasthuis Haarlem.

¹⁶For this study, major colorectal surgery was defined as a resection of the colon or rectum with reconstruction via ostomy or anastomosis (or both).

Surgical procedure

Patients underwent one of: subtotal colectomy, hemicolectomy, sigmoid resection, low anterior resection, or abdominoperineal resection. Standardized antibiotic prophylaxis¹⁷ was used for all procedures. Patients were treated postoperatively according to local protocol, consisting of admittance to either the intensive-care unit, medium care, or general ward.

Control and AL groups

Control group The control group consisted of 27 patients admitted to VUmc who did not develop AL in the postoperative period. Patients admitted to VUmc were recruited prior to surgery, and urine samples were collected on the third postoperative day, with patients excluded from the control group if they developed AL.

AL group The AL group consisted of 22 patients diagnosed with postoperative AL through clinical deterioration and radiological imaging or surgical intervention. In cases of certain AL, urine samples were collected immediately.

Sample handling and FAIMS analysis

Samples were collected in 4.5 ml CryoPure containers and frozen at -80°C within two hours of collection. Samples were stored for a maximum of nine months to minimize VOC degradation [Esfahani et al., 2016]. Samples were analysed in batches on a Lonestar FAIMS device (Owlstone, UK). Samples were removed from the freezer a maximum of two hours prior to analysis and allowed to defrost at room temperature. 4.5 ml aliquots of each sample were then placed in 22 ml glass vials before being introduced into the Lonestar and heated to 38°C . Three FAIMS analysis runs were performed for each sample. Blank samples consisting of tap water were run three times before and after each urine sample to flush any traces of previous samples from the device and return the sensors to baseline response.

Statistical analysis

For this study, we compared a new approach to the standard FAIMS analysis used in the previous AL study described in section 2.4.4. For the standard analysis, we used Random Forests with the 64 top features retained, the model selected in the previous study. For details of the methodology, please refer to section 2.4.4. For

¹⁷Cefuroxime or metronidazole.

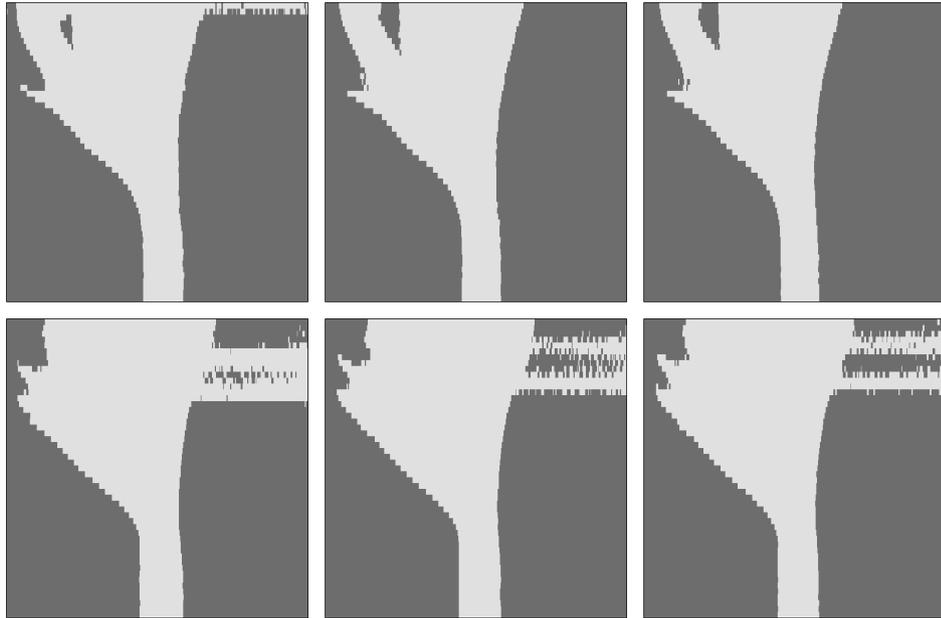


Figure 2.24: Variables excluded by standard deviation thresholding for the three FAIMS runs. The top row shows the positive ion stream and the bottom row shows the negative ion stream. Variables in light grey are included in the analysis, and variables in dark grey are excluded.

comparison to the previous study, we also ran analyses using the models in Table 2.6, the results for which are shown in Table 2.9.¹⁸

The new approach uses a considerably simplified pipeline. First, noise variables are excluded by standard deviation thresholding as described in section 2.2.3, with 65% of variables removed as noise. Which variables were retained and excluded can be seen in Figure 2.24. The retained variables are then passed as input to a Random Forest classifier. This removes both the wavelet transformation and the variable selection step from the standard approach, relying instead on the feature selection inherent to Random Forests. As with the wavelet transformation in the standard pipeline, the standard deviation thresholding is unsupervised and therefore is done before the cross-validation is performed.

Results

Sensitivity, specificity, and ROC AUC for the two analysis pipelines are shown in Table 2.8, and the associated ROC curves can be seen in Figure 2.25. Random

¹⁸With the exception of stochastic gradient boosting, which failed to run due to the sample size being too small for `caret` to perform hyperparameter tuning.

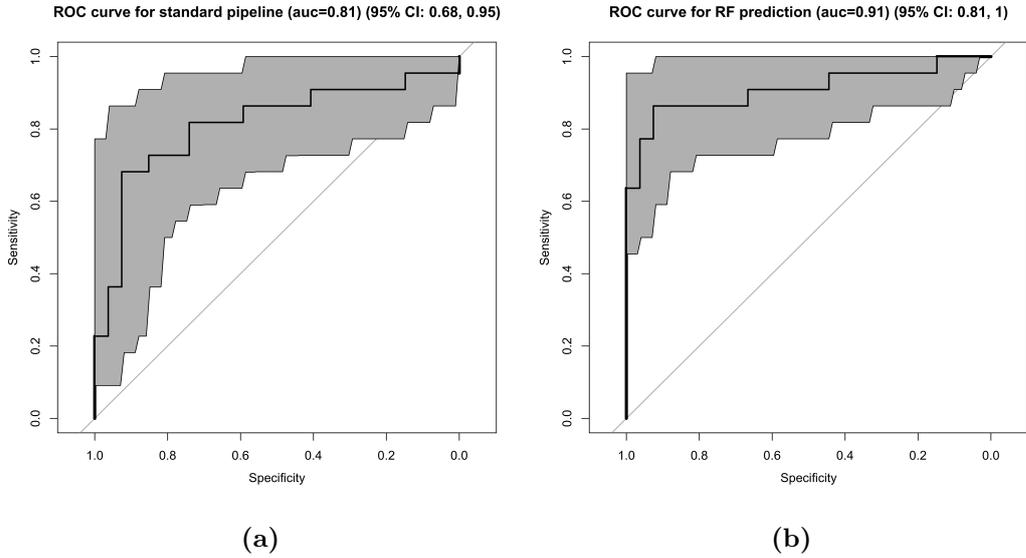


Figure 2.25: ROC curves for detection of AL after major colorectal surgery. **(a)** The standard analysis pipeline utilizing wavelet transformation, feature selection using MWW, and classification using Random Forests. **(b)** Removal of noise variables using standard deviation thresholding and classification by Random Forests.

Forests with standard deviation thresholding significantly outperformed the standard pipeline at the 97.5% confidence level based on a bootstrapped difference between AUCs, with a 95% confidence interval for the difference of (0.017, 0.23).

For comparison with the previous study, Table 2.9 recreates the model selection analysis for the first AL study, as seen in Table 2.6. All algorithms performed similarly well, unlike the previous study where some algorithms performed consistently worse than others.

Discussion

In this study we found further evidence that urinary VOCs can be used as a diagnostic test for post-operative anastomotic leakage. The strong performance across all tested classifiers is encouraging, and suggests that the classification problem is robust to the specific classifier used. Performance was comparable to that found in the previous AL study, and has the benefit that the number of patients involved in this study is significantly higher (49 vs 32) with a more balanced distribution of classes (45% positive vs 19% positive).

	<i>AUC</i>	<i>Sensitivity</i>	<i>Specificity</i>
<i>Standard analysis</i>	0.81 (0.68, 0.95)	0.86 (0.65, 0.96)	0.59 (0.39, 0.77)
<i>Thresholding and Random Forest</i>	0.91 (0.81, 1.0)	0.86 (0.65, 0.96)	0.93 (0.75, 1.0)

Table 2.8: Results for training a Random Forest classifier on urinary VOC data to detect anastomotic leakage after major colorectal surgery, using both the standard analysis with wavelet transformation, and the simplified analysis using thresholding. 95% confidence intervals are shown in brackets. There is a statistically significant difference (at the 5% level) between the AUCs of the two methods, with a 95% confidence interval for the difference of (0.017, 0.23).

A key weakness of this study is the disparity between sample collection methods for the AL-positive and AL-negative groups. AL-negative samples were collected on the third postoperative day, whereas AL-positive samples were collected immediately after diagnosis of AL. Immediate collection after diagnosis mitigates any effects of detecting the treatment for AL, but is still likely to introduce a positive bias to the results.

The improved performance in the simplified analysis pipeline raises interesting questions about the utility of the standard analysis pipeline. Anecdotally, for other studies when similar approaches were used the standard pipeline using wavelet transformation and feature selection outperformed more generic approaches. One hypothesis for the results in this case is that the standard approach is focussed on de-noising the data, boosting the signal-to-noise ratio. This approach works well when the signal-to-noise ratio is low, as in other studies with more challenging classification problems, but in this study with a clear signal separating the two classes the de-noising efforts are counter-productive.

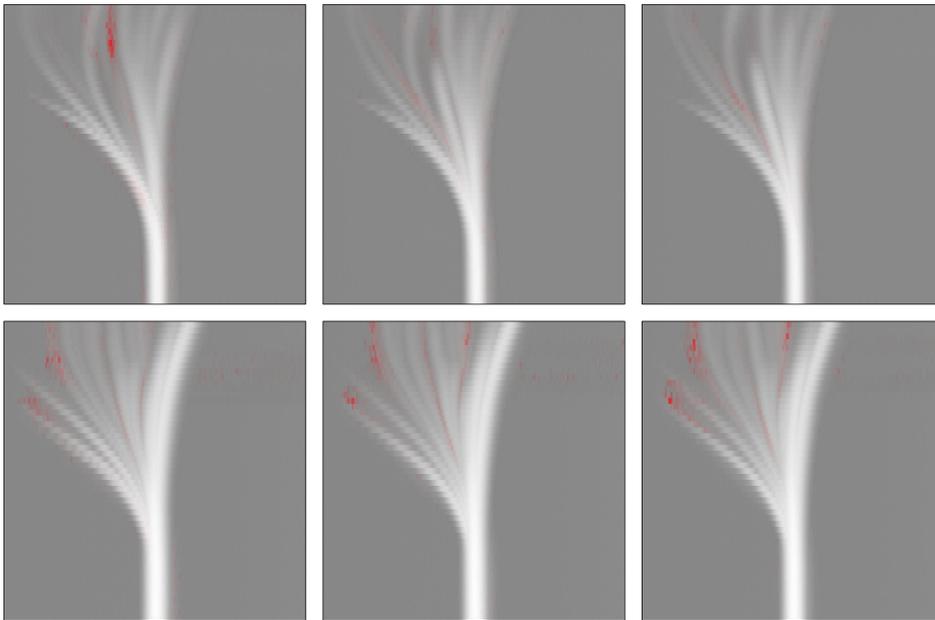


Figure 2.26: Variable importance for Random Forest classification of anastomotic leakage after major colorectal surgery. VOC plumes present in the dataset are shown in white, determined by the maximum ion count achieved for each variable. Important variables are plotted in red, with stronger red colouring indicating higher importance.

Model	# features retained	AUC (95% CI)
Random Forest	16	0.77* (0.64, 0.91)
Sparse Logistic Regression	16	0.88* (0.78, 0.98)
SVM (radial kernel)	16	0.78* [†] (0.64, 0.92)
SVM (linear kernel)	16	0.86* (0.75, 0.98)
Random Forest	64	0.81* (0.68, 0.95)
Sparse Logistic Regression	64	0.87* (0.76, 0.98)
SVM (radial kernel)	64	0.84* (0.71, 0.96)
SVM (linear kernel)	64	0.86* (0.74, 0.97)
Random Forest	128	0.85* (0.75, 0.96)
Sparse Logistic Regression	128	0.82* (0.70, 0.94)
SVM (radial kernel)	128	0.85* (0.74, 0.97)
SVM (linear kernel)	128	0.88* (0.78, 0.98)

Table 2.9: Results for the standard analysis pipeline for detection of AL after major colorectal surgery. Statistically significant results (at a 5% level) are marked with an asterisk (*). Results which have a statistically significant (at the 5% level) difference in AUC to the model with the highest AUC (sparse logistic regression with 64 features) are marked with a dagger (†).

Chapter 3

Gaussian Process Latent Variable Models - Theory and Literature Review

3.1 Overview

Dimensionality reduction provides a lower-dimension representation of a data set whilst retaining as much information as possible relevant to the problem at hand. Dimensionality reduction is motivated by the observation that high-dimensional data are often highly redundant, and consolidating the information in the data into a smaller number of latent variables can improve the ability to visualize it, reduce the computational cost of subsequent algorithms, and reduce the risk of overfitting through variable selection.

A Gaussian process latent variable model (GPLVM) is an unsupervised dimensionality reduction technique which treats the observed variables as a function of latent variables in some lower-dimensional latent space, with the function mapping from the latent to the observed space being drawn from a Gaussian process [Lawrence, 2004]. The locations of the data in latent space are then treated as parameters of the model and optimized with respect to the likelihood of the model to determine a lower-dimensional representation of the observed data.

In this section we review the theory of Gaussian processes and GPLVMs, and discuss recent developments in Gaussian process and GPLVMs methodology.

3.2 Background Theory

3.2.1 Gaussian processes

As the name suggests, a GP latent variable model approaches the relationship between the latent and observed spaces by modelling it using GPs. In this section we present a brief introduction to the theory of GPs. For a more detailed treatment we can heartily recommend the excellent book by Rasmussen and Williams [2006] on the subject, from which the results in this section are drawn.¹

GPs can be formulated in two ways which result in equivalent outcomes. On the one hand, given some observed data, GPs can be seen as performing Bayesian linear regression after the input points have been projected to a higher dimensional “feature space”, allowing for a significantly more expressive model than standard Bayesian linear regression. This is facilitated by the use of the “kernel trick”, where inner products between vectors projected into a feature space can be computed implicitly using a positive definite kernel function² over the input space. This avoids the need to explicitly work in the feature space and therefore potentially saves significantly on computational cost. Kernel methods are a key part of a number of popular machine learning techniques [Hofmann et al., 2008], notably SVMs where they make non-linear modelling computationally feasible. Gaussian processes combine the computational benefits and non-linearity of kernel methods with a principled Bayesian framework for prediction and hyperparameter optimization.

In the other formulation, we consider a probability distribution over functions. A GP can be used as a prior over functions which could generate the observed data, from which we derive a posterior distribution over functions conditioned on the observed data. The kernel in this formulation specifies the class of functions which the prior encodes for, and loosely speaking specifies the prior covariance between points in the input space.

The link between the kernel function and the feature space it implicitly projects into is made clear by the Moore–Aronszajn theorem, which states that for every

¹It is worth noting that Rasmussen and Williams [2006] use the convention that the design matrix X is structured with variables along rows and samples along columns, the transpose of the usual convention in statistical textbooks. In the following we use the convention that the design matrix has variables along the columns and samples along the rows, and so some formulae will have superficial differences to those presented in Rasmussen and Williams [2006].

²A symmetric function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a positive definite kernel function iff for all finite $X = \{x_1, x_2, \dots, x_n\} \subset \mathcal{X}$ the Gram matrix \mathbf{K} constructed by pairwise evaluation of k over the elements of X , $\mathbf{K}_{ij} = k(x_i, x_j)$, is positive definite.

symmetric positive definite kernel function there is a unique reproducing kernel Hilbert space (RKHS) for which that kernel is the reproducing kernel. We explain this link in more detail in Section 3.2.2.

Linear regression in feature space

In Bayesian linear regression, one models the generating function of observed data as a linear function of the inputs \mathbf{x} with Gaussian additive noise ϵ , placing a Gaussian prior over the linear function weights \mathbf{w} :

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \quad (3.1)$$

$$y = f(\mathbf{x}) + \epsilon \quad (3.2)$$

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2) \quad (3.3)$$

$$\mathbf{w} \sim \mathcal{N}(0, \Sigma_p) \quad (3.4)$$

Given observed data $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T$ and \mathbf{y} , the posterior for this model is:

$$p(\mathbf{w} \mid \mathbf{X}, \mathbf{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2} \mathbf{A}^{-1} \mathbf{X}^T \mathbf{y}, \mathbf{A}^{-1}\right) \quad (3.5)$$

$$\mathbf{A} = \frac{1}{\sigma_n^2} \mathbf{X}^T \mathbf{X} + \Sigma_p^{-1} \quad (3.6)$$

and given an unobserved point \mathbf{x}_* , the predictive distribution for $f_* = f(\mathbf{x}_*)$, is:

$$f_* \mid \mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}\left(\frac{1}{\sigma_n^2} \mathbf{x}_*^T \mathbf{A}^{-1} \mathbf{X}^T \mathbf{y}, \mathbf{x}_*^T \mathbf{A}^{-1} \mathbf{x}_*\right) \quad (3.7)$$

To increase the expressiveness of the model, we would like to first project \mathbf{x} into a higher-dimensional feature space before performing a linear regression. In general, let ϕ be a function that maps points in the input space into an N -dimensional feature space. We then have:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \quad (3.8)$$

Using the notation $\Phi(\mathbf{X})$ for the row-wise concatenation of $\phi(\mathbf{x})$ for the columns \mathbf{x} of \mathbf{X} , i.e. $\Phi(\mathbf{X}) = (\phi(\mathbf{x}_1), \dots, \phi(\mathbf{x}_n))^T$, we can substitute $\Phi(\mathbf{X})$ and $\phi(\mathbf{x}_*)$ into equation 3.7 to find the predictive distribution for f_* in this case:

$$f_* \mid \phi(\mathbf{x}_*), \Phi(\mathbf{X}), \mathbf{y} \sim \mathcal{N}\left(\frac{1}{\sigma_n^2} \phi(\mathbf{x}_*)^T \mathbf{A}^{-1} \Phi(\mathbf{X})^T \mathbf{y}, \phi(\mathbf{x}_*)^T \mathbf{A}^{-1} \phi(\mathbf{x}_*)\right) \quad (3.9)$$

where \mathbf{A} is now

$$\mathbf{A} = \frac{1}{\sigma_n^2} \Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \Sigma_p^{-1} \quad (3.10)$$

Equation 3.9 can be re-written to enable us to use the ‘kernel trick’, which allows us to avoid the need to explicitly compute the mapping of our inputs into feature space by writing our problem in terms of inner products in the feature space, which can in turn be written as positive-definite functions over the input space:

$$\begin{aligned} f_* \mid \phi(\mathbf{x}_*), \Phi(\mathbf{X}), \mathbf{y} &\sim \mathcal{N}(\mu_*, \sigma_*^2) \\ \mu_* &= \phi(\mathbf{x}_*)^T \Sigma_p \Phi(\mathbf{X})^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \\ \sigma_*^2 &= \phi(\mathbf{x}_*)^T \Sigma_p \phi(\mathbf{x}_*) - \phi(\mathbf{x}_*)^T \Sigma_p \Phi(\mathbf{X})^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \Phi(\mathbf{X})^T \Sigma_p \phi(\mathbf{x}_*) \\ \mathbf{K} &= \Phi(\mathbf{X}) \Sigma_p \Phi(\mathbf{X})^T \end{aligned} \quad (3.11)$$

For the mean, consider that

$$\begin{aligned} \frac{1}{\sigma_n^2} \Phi(\mathbf{X})^T (\mathbf{K} + \sigma_n^2 \mathbf{I}) &= \frac{1}{\sigma_n^2} \Phi(\mathbf{X})^T (\Phi(\mathbf{X}) \Sigma_p \Phi(\mathbf{X})^T + \sigma_n^2 \mathbf{I}) \\ &= \left(\frac{1}{\sigma_n^2} \Phi(\mathbf{X})^T \Phi(\mathbf{X}) + \Sigma_p^{-1} \right) \Sigma_p \Phi(\mathbf{X})^T \\ &= \mathbf{A} \Sigma_p \Phi(\mathbf{X})^T \end{aligned} \quad (3.12)$$

Then right-multiplying by $(\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}$ and left-multiplying by \mathbf{A}^{-1} gives

$$\frac{1}{\sigma_n^2} \mathbf{A}^{-1} \Phi(\mathbf{X})^T = \Sigma_p \Phi(\mathbf{X})^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \quad (3.13)$$

and μ_* follows from substituting this into equation 3.9. For σ_*^2 , apply Woodbury’s matrix identity³ to \mathbf{A}^{-1} :

$$\begin{aligned} \mathbf{A}^{-1} &= \left(\Sigma_p^{-1} + \frac{1}{\sigma_n^2} \Phi(\mathbf{X})^T \Phi(\mathbf{X}) \right)^{-1} \\ &= \Sigma_p - \Sigma_p \Phi(\mathbf{X})^T \left(\sigma_n^2 + \Phi(\mathbf{X}) \Sigma_p \Phi(\mathbf{X})^T \right)^{-1} \Phi(\mathbf{X}) \Sigma_p \\ &= \Sigma_p - \Sigma_p \Phi(\mathbf{X})^T (\mathbf{K} + \sigma_n^2)^{-1} \Phi(\mathbf{X}) \Sigma_p \end{aligned} \quad (3.14)$$

and again substitute into equation 3.9.

³ $(\mathbf{B} + \mathbf{UCV})^{-1} = \mathbf{B}^{-1} - \mathbf{B}^{-1} \mathbf{U} (\mathbf{C}^{-1} + \mathbf{VB}^{-1} \mathbf{U})^{-1} \mathbf{VB}^{-1}$, see appendix D.1.1 for a proof.

Given a kernel function $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\Sigma_p\phi(\mathbf{x}')$, we can now compute the parameters for equation 3.9 without the need to work in our feature space explicitly, avoiding the need to compute projected input points and also allowing us to invert \mathbf{K} rather than the (normally) considerably larger \mathbf{A} . Indeed, for many standard kernels the dimension of the associated feature space is infinite. We will see that this treatment of Bayesian linear regression after projecting into feature space is equivalent to the definition of GPs below as a probability distribution over functions, with a GP as a prior with covariance function k as above and mean function $m(\mathbf{x}) = 0$. The posterior obtained by conditioning on the points used for regression then gives equivalent results to those given above.

A probability distribution over functions

An alternative view of GPs is as a probability distribution over functions. Formally, we define a Gaussian process as a collection of random variables, any finite number of which have a joint Gaussian distribution. GPs are defined over a (usually infinite⁴) index set \mathcal{X} , which corresponds with the input values for the functions the distribution covers. A GP is specified by its mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, $\mathbf{x} \in \mathcal{X}$ and its covariance function $k(\mathbf{u}, \mathbf{v}) = \mathbb{E}[(f(\mathbf{u}) - m(\mathbf{u})) (f(\mathbf{v}) - m(\mathbf{v}))]$, $\mathbf{u}, \mathbf{v} \in \mathcal{X}$. We write this GP as

$$f \sim \mathcal{GP}(m, k) \tag{3.15}$$

and, for a finite subset $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ of \mathcal{X} we have from the definition of GPs

$$\mathbf{f}_X \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}(X, X)) \tag{3.16}$$

where \mathbf{f}_X and \mathbf{m}_X are vectors whose i^{th} entries are $f(\mathbf{x}_i)$ and $m(\mathbf{x}_i)$ respectively, and $\mathbf{K}(X, X)$ is a matrix whose $(i, j)^{\text{th}}$ entry is $k(\mathbf{x}_i, \mathbf{x}_j)$. In general, we will assume $m(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathcal{X}$, since we will be using GPs as priors and this restriction does not restrict the mean of the posterior to also be identically zero.

The definition of a GP given above implies a consistency requirement (or marginalisation property). That is, for $U \subset V \subseteq \mathcal{X}$, marginalising out the variables associated with $V \setminus U$ from the distribution of \mathbf{f}_V should result in an identical distribution to that specified by the GP for \mathbf{f}_U . Put another way, considering the value of f at additional points should not change the distribution of the points already under consideration. This requirement is automatically fulfilled if the covariance function

⁴For a finite index set, a GP is equivalent to a joint Gaussian distribution over the values of f at the points in the index set.

k specifies the entries in the covariance matrix $\mathbf{K}(X, X)$.

Prediction Given noisy observed values at points \mathbf{X} with values \mathbf{y} , and points \mathbf{X}_* at which we wish to predict \mathbf{f}_* , the joint distribution for \mathbf{y} and \mathbf{f}_* under a GP prior is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix} \right) \quad (3.17)$$

where $[\mathbf{K}(\mathbf{A}, \mathbf{B})]_{ij} = k(\mathbf{A}_{i*}, \mathbf{B}_{j*})$ is the covariance between the i^{th} point in \mathbf{A} and the j^{th} point in \mathbf{B} , and we have assumed i.i.d. Gaussian noise with mean zero and variance σ_n^2 on the observed values. Conditioning this joint distribution on the observed values gives the predictive distribution for \mathbf{f}_*

$$\begin{aligned} \mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N} \left(\mathbf{K}(\mathbf{X}_*, \mathbf{X}) \left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} \right)^{-1} \mathbf{y}, \right. \\ \left. \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) - \mathbf{K}(\mathbf{X}_*, \mathbf{X}) \left(\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} \right)^{-1} \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \right) \end{aligned} \quad (3.18)$$

To obtain the predictive distribution for the noisy observations \mathbf{y}_* , one can add the noise variance σ_n^2 to the predictive variance of \mathbf{f}_* .

3.2.2 Reproducing kernel Hilbert spaces - a brief foray into functional analysis

The theory of the RKHS gives a useful insight into how GPs and the kernel trick operate in feature space. Rather than a vague explanation about implicit projections into higher-dimensional feature spaces accompanied by judicious hand-waving, we can (through a constructive proof of the Moore-Aronszajn theorem) provide an explicit description of the feature space associated with a particular kernel function and also the projections of our input space into it. A more detailed treatment of the material covered here is available in Appendix D.2.

First, some brief definitions. A Hilbert space is a real or complex inner product space which is complete with respect to the norm induced by the inner product.⁵

⁵ \mathbb{R}^n equipped with the dot product is a straightforward example of a Hilbert space, but Hilbert spaces extend to far more exotic fare. A common (and only slightly more exotic) example is the infinite-dimensional Lebesgue space $L^2(\mathbb{R})$, the space of square-integrable complex functions on \mathbb{R} , equipped with the inner product $\langle f, g \rangle = \int_{\mathbb{R}} f(x) \overline{g(x)} dx$. f is square-integrable on \mathbb{R} iff $\int_{\mathbb{R}} |f(x)|^2 dx < \infty$.

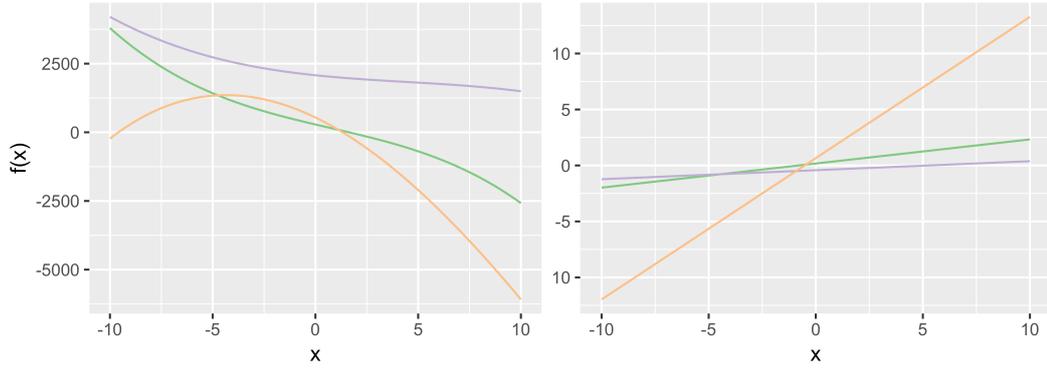


Figure 3.1: Examples of functions drawn from GP priors with **(left)** a polynomial kernel of degree $p = 3$, and **(right)** a linear kernel with $\sigma_0^2 = 1$.

A reproducing kernel Hilbert space (RKHS) is a Hilbert space H of real-valued functions on an arbitrary index set \mathcal{X} with a function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that:

1. $K_x: x' \mapsto k(x', x) \in H \forall x \in \mathcal{X}$
2. k has the reproducing property: $\langle f, K_x \rangle_H = f(x) \forall x \in \mathcal{X}, f \in H$

The Moore-Aronszajn theorem states that for every positive definite kernel function k there is a unique RKHS for which k is the reproducing kernel, that is, there is a bijection between the set of positive definite kernel functions and the set of reproducing kernel Hilbert spaces. From the constructive proof of the Moore-Aronszajn theorem (presented in Appendix D.2.3) we can see that the implicit feature space associated with a kernel $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is the completion of the linear span of the set of functions $\{K_x: x' \mapsto k(x, x') \mid x \in \mathcal{X}\}$, and the link between this formulation and the view of GPs as implicit Bayesian linear regression in a feature space is made clear, since now the implicit projection of a point $x \in \mathcal{X}$ into the feature space associated with some kernel function k can be seen to be the function $K_x: x' \mapsto k(x, x')$, with the inner product $\langle K_x, K_{x'} \rangle = k(x, x')$.

3.2.3 Kernel functions

As we saw in section 3.2.2, the kernel function k defines the function space a Gaussian process operates in, and as such specifies the type of functions the GP prior encodes for. In this section we review a number of common kernel functions and the function spaces they specify.

Linear and polynomial kernels

The linear kernel [Rasmussen and Williams, 2006] is defined as

$$k_{\text{lin}}(\mathbf{x}, \mathbf{x}') = \sigma_0^2 + \mathbf{x} \cdot \mathbf{x}' \quad (3.19)$$

and more generally the polynomial kernel is defined as

$$k_p(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^p \quad (3.20)$$

The linear kernel is equivalent to Bayesian linear regression with $\mathcal{N}(0, 1)$ priors on the coefficients of x_d and a prior of $\mathcal{N}(0, \sigma_0^2)$ on the bias [Rasmussen and Williams, 2006, p. 80]. The polynomial kernel k_p regresses over polynomial functions of order p .

Squared exponential kernel

The squared exponential kernel encodes a prior over smooth functions, where the length-scale l controls how distant two points must be before their corresponding function values are uncorrelated. The squared exponential kernel has the formula:

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right) \quad (3.21)$$

Automatic relevance determination (ARD) kernel

The concept of automatic relevance determination was first introduced in the context of neural networks [MacKay et al., 1994; Neal, 1995; Bishop, 2006]. The ARD kernel is a natural extension of the squared exponential kernel, where instead of one length-scale l shared between all dimensions in the input space, the i^{th} dimension is associated with length-scale l_i . This has two notable benefits. First, if the variables in your input space are not commensurate, the ARD kernel will prevent the variable with the largest variance from dominating the covariance function by automatically scaling each dimension. Second, the ARD kernel can “turn off” uninformative input variables by setting the associated length-scales to be large. It is this latter property which gives the ARD kernel its name. The ARD kernel has the formula

$$k_{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\sum_i \frac{(x_i - x'_i)^2}{2l_i^2}\right) \quad (3.22)$$

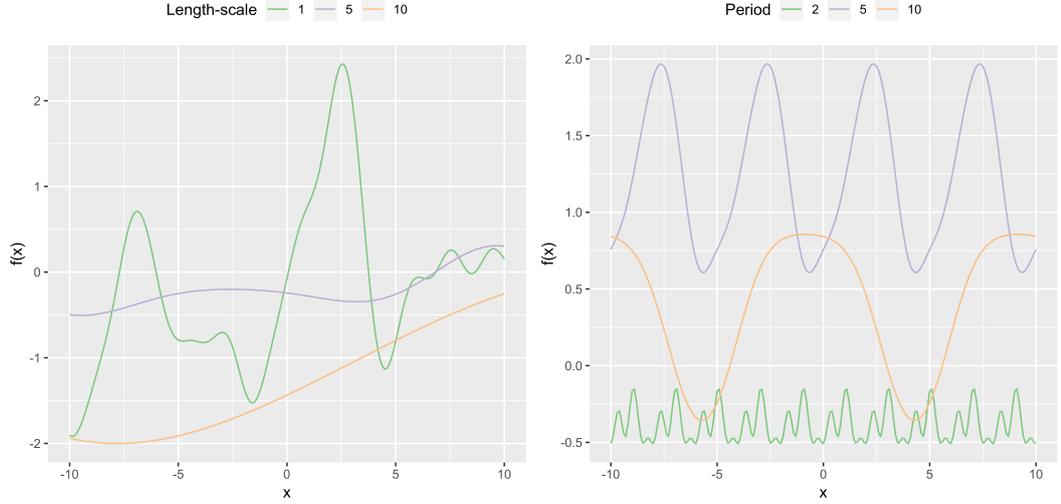


Figure 3.2: Examples of the effect of varying **(left)** the length-scale of the squared exponential kernel, and **(right)** the period of the periodic kernel.

For the purposes of sparse optimisation and interpretability of the hyperparameters, it is often preferable to optimise with respect to the inverse length-scales, $\lambda_i = l_i^{-1}$, so non-informative variables' inverse length-scales are set either to zero or close to zero:

$$k_{\text{ARD}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2} \sum_i \lambda_i^2 \cdot (x_i - x'_i)^2\right) \quad (3.23)$$

Rational quadratic kernel

The rational quadratic kernel is a scale mixture of squared exponential kernels of different length-scales, with the mixing distribution being a gamma distribution over the inverse squared length-scales of the constituent squared exponential distributions $\tau = l_{\text{SE}}^{-2}$:

$$p(\tau \mid \alpha, \beta) \propto \tau^{\alpha-1} \exp\left(\frac{-\alpha\tau}{\beta}\right) \quad (3.24)$$

where $\beta = l^{-2}$.

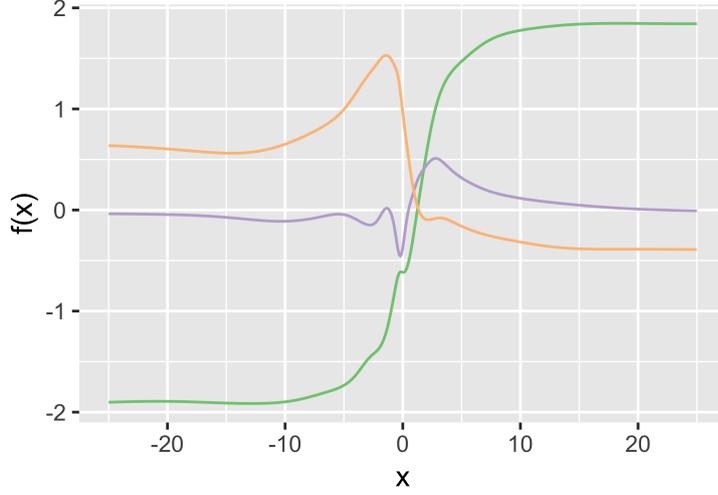


Figure 3.3: Examples of functions sampled from a GP prior with a neural network kernel with $\Sigma = \mathbf{I}$.

Periodic kernel

The periodic kernel has the formula

$$k_{\text{per}}(\mathbf{x}, \mathbf{x}') = \exp\left(-2l^{-2} \sin^2\left(\frac{\pi|\mathbf{x} - \mathbf{x}'|}{p}\right)\right) \quad (3.25)$$

and models functions which repeat exactly with period p . As with the squared exponential kernel, l determines the length-scale of the function, with a lower value of l allowing for faster variation as \mathbf{x} changes.

Neural network kernel

With appropriate kernel choices, GPs can be used to model other machine learning algorithms. The neural network kernel [Williams, 1997] models a neural network with one hidden layer of N_H nodes as N_H tends to infinity, with the transfer function $\Phi(z) = 2/\sqrt{\pi} \int_0^z \exp(-t^2) dt$ and hidden node values computed using $h(\mathbf{x}; \mathbf{u}) = \Phi(u_0 + \sum_{j=1}^D u_j x_j)$, $\mathbf{u} \sim \mathcal{N}(0, \Sigma)$. It has the formula

$$k_{NN}(\mathbf{x}, \mathbf{x}') = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}'}}{\sqrt{(1 + 2\tilde{\mathbf{x}}^T \Sigma \tilde{\mathbf{x}})(1 + 2\tilde{\mathbf{x}}'^T \Sigma \tilde{\mathbf{x}'})}}\right) \quad (3.26)$$

where $\tilde{\mathbf{x}} = (1, x_1, \dots, x_n)$ is an augmented input vector whose first entry corresponds to the bias.

Random partition kernels

Random partition kernels [Davies and Ghahramani, 2014] represent a novel class of kernels which, in contrast to the previously discussed kernels which are analytically derived, are based on random partitioning of the data. Examples of random partitions are those produced by ensemble tree methods such as Random Forests, or by stochastic clustering algorithms. For example, for Random Forests, we can consider each tree in the forest as a partition of the input space X , with the clusters for an element x of X for a given tree being the leaf node that x maps to. This gives a supervised kernel with no hyperparameters to optimize, which can allow a significant speed boost for large datasets.

First, some definitions. A partition of a dataset D is a set of non-empty, pairwise-disjoint subsets of D which form a cover of D , that is, P is a partition of D iff:

$$P = \{C_i \mid i \in \{1, \dots, n\}, C_i \subset D, C_i \neq \emptyset\} \quad (3.27)$$

$$C_i \cap C_j = \emptyset \quad \forall i \neq j \quad (3.28)$$

$$\bigcup_{i=1}^n C_i = D \quad (3.29)$$

Each C_i is called a cluster.

Let $P_D \subset \mathcal{P}(\mathcal{P}(D))$ be the set of all possible partitions of D . We can define a function $\phi: D \times P_D \rightarrow \mathbb{N}$ which, given an element x of D and a partition P of D , returns the index of the cluster in P which contains x . ϕ is defined for all $x \in D$ since P is a cover of D , and is well-defined since the clusters are pairwise disjoint.

A partition distribution \mathcal{Q} over the partitions of D is a pdf which assigns a probability to each possible partition of D . P is called a random partition if it is drawn from a partition distribution.

Now, we can define the random partition kernel. Given a partition distribution \mathcal{Q} over our input set X , define the kernel

$$k_{\mathcal{Q}}(\mathbf{x}, \mathbf{x}') = \mathbb{E} \left[I \left[\phi(\mathbf{x}, P) = \phi(\mathbf{x}', P) \right] \right]_{P \sim \mathcal{Q}} \quad (3.30)$$

to be the random partition kernel induced by \mathcal{Q} , where I is the indicator function.

In many practical situations the kernel as defined above is not practical to evaluate analytically. Luckily, we can construct an approximation based on a sample from \mathcal{Q} . For a sample $Q = \{P_i \mid i \in \{1, \dots, m\}, P_i \sim \mathcal{Q}\}$ from \mathcal{Q} of size m , we have:

$$k_{\mathcal{Q}}(\mathbf{x}, \mathbf{x}') \approx \frac{1}{m} \sum_{i=1}^m I[\phi(\mathbf{x}, Q_i) = \phi(\mathbf{x}', Q_i)] \quad (3.31)$$

that is, the covariance between \mathbf{x} and \mathbf{x}' is the fraction of times they are assigned to the same cluster over the m samples.

Combining kernels

For two positive-definite kernel functions k_1 and k_2 over some space \mathcal{X} and $\alpha \in \mathbb{R}_{>0}$, the following are also positive-definite kernel functions:

$$\alpha k_1 \quad (3.32)$$

$$k_1 + k_2 \quad (3.33)$$

$$k_1 \cdot k_2 \quad (3.34)$$

Positive definiteness of 3.32 and 3.33 are easy to see by considering Gram matrices $\mathbf{K}_1, \mathbf{K}_2$ derived from k_1 and k_2 . For 3.32, $\mathbf{z}^* \alpha \mathbf{K}_1 \mathbf{z} = \alpha \mathbf{z}^* \mathbf{K}_1 \mathbf{z} \geq 0$ with equality iff $\mathbf{z} = \mathbf{0}$. For 3.33, $\mathbf{z}^* (\mathbf{K}_1 + \mathbf{K}_2) \mathbf{z} = \mathbf{z}^* \mathbf{K}_1 \mathbf{z} + \mathbf{z}^* \mathbf{K}_2 \mathbf{z} \geq 0$, again with equality iff $\mathbf{z} = \mathbf{0}$.

For 3.34, note that the Gram matrix of $k_1 \cdot k_2$ is the Hadamard product of \mathbf{K}_1 and \mathbf{K}_2 , and we use the Schur product theorem [Schur, 1911]. This states that the Hadamard product of two positive-definite matrices is itself positive-definite. A simple proof of this theorem is described in Styan [1973], and is as follows. Given positive-definite \mathbf{A} and \mathbf{B} and some suitably-sized vector \mathbf{z} , consider $\mathbf{z}^* (\mathbf{A} \circ \mathbf{B}) \mathbf{z}$, where $\mathbf{A} \circ \mathbf{B}$ is the Hadamard product of \mathbf{A} and \mathbf{B} . Since \mathbf{B} is positive-definite, it has a Cholesky decomposition $\mathbf{B} = \mathbf{L}\mathbf{L}^*$. Substituting this into the quadratic form gives

$$\mathbf{z}^* (\mathbf{A} \circ \mathbf{B}) \mathbf{z} = \mathbf{z}^* (\mathbf{A} \circ (\mathbf{L}\mathbf{L}^*)) \mathbf{z} \quad (3.35)$$

$$= \sum_{i,j=1}^n z_i a_{ij} \left(\sum_{k=1}^n l_{ik} l_{jk} \right) z_j \quad (3.36)$$

$$= \sum_{k=1}^n \left(\sum_{i,j=1}^n z_i l_{ik} a_{ij} z_j l_{jk} \right) \quad (3.37)$$

$$= \sum_{k=1}^n (\mathbf{z} \circ \mathbf{l}_k)^* \mathbf{A} (\mathbf{z} \circ \mathbf{l}_k) \quad (3.38)$$

$$\geq 0 \quad (3.39)$$

where \mathbf{l}_k is the k 'th column of \mathbf{L} . Since \mathbf{A} is positive-definite and $\mathbf{B} \neq \mathbf{0} \implies \mathbf{L} \neq \mathbf{0}$, equality occurs iff $\mathbf{z} = \mathbf{0}$, and so $\mathbf{A} \circ \mathbf{B}$ is positive-definite.

Combining kernels allows for tremendous flexibility when using GPs for regression. In practice, a kernel k is almost always scaled by a hyperparameter α to allow the variance of the regressed function to be optimized.

3.2.4 Hyperparameter optimization

Background

Most machine learning algorithms have hyperparameters which control how the algorithm models a problem, for example the choice of k in the k -NN algorithm. Varying the hyperparameters can have a dramatic effect on the efficacy of a model; some models (such as SVM) are sensitive to the choice of hyperparameters, while other models (such as Random Forests) are robust to the choice of hyperparameter [Caruana and Niculescu-Mizil, 2006], often performing close to optimum using “default” hyperparameters [Svetnik et al., 2004].

Methods for optimally selecting hyperparameters constitute a varied and active field of research, and can be considered to be a subset of model selection [Gold and Sollich, 2003; Bergstra et al., 2013]. Excessive tuning of model hyperparameters can lead to the model performing well on the training data, but generalizing badly when presented with new data. On the other hand, insufficient tuning runs the risk of causing a model to perform significantly worse than the optimum performance which it could achieve [Cawley and Talbot, 2010]. This balancing act reflects the classic bias-variance dilemma [Geman et al., 1992], a leitmotif of machine learning, where increasing the complexity of a model improves its accuracy (reducing its bias) by increasing how expressive the model is, but in doing so also increases the sensitivity of the model’s output to the specific sample of training data used to fit it (increasing the model’s variance over multiple training set samples). In the context of hyperparameter tuning, an increasingly thorough search of hyperparameter space reduces the bias of the model while increasing its variance.

A wide range of approaches are available for hyperparameter selection which are not model-specific. A common and straightforward technique is grid search [Hsu et al., 2003], where the algorithm’s performance is examined on points forming a grid in hyperparameter space. For each hyperparameter α_i a set of values A_i is specified, and the model performance is evaluated at each point in $A_1 \times A_2 \times \dots \times A_n$. Performance at each point is usually assessed using cross-validation or a hold-out test set to combat over-fitting. Increasing the number of points in the grid increases the likelihood of finding good hyperparameters at the cost of increased computational complexity and risk of over-fitting. A similar and often more efficient approach is random search [Bergstra and Bengio, 2012], where hyperparameter values are randomly and independently sampled for testing. Random search is based on the observation that frequently only a few hyperparameters will have a significant impact on performance, and so random search will be able to test more values of the relevant hyperparameters for the same cost as grid search.

Grid search and random search are simple and quick to implement, but if a model is costly to train the time to examine sufficient hyperparameter values to achieve a satisfactory result may be prohibitive. One more sophisticated method of particular note is Bayesian optimization [Eggenberger et al., 2013], where a probabilistic model of the performance of a machine learning algorithm is conditioned on point evaluations of the model performance, with the posterior distribution being used to select the next set of hyperparameters to evaluate and eventually also to predict the optimum hyperparameter settings. Many different models have been used for Bayesian optimization [Thornton et al., 2013; Bergstra et al., 2013, 2011], including using GPs [Snoek et al., 2012].

Hyperparameter optimization for Gaussian processes

The formulae of many kernels used for GPs contain hyperparameters, and these will normally need to be tuned to optimize the fit of the model to the data. A benefit GPs have over many other machine learning algorithms is that there is a natural, principled method for optimizing the kernel hyperparameters $\boldsymbol{\theta}$ based on the marginal likelihood $p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta})$ of the model, where we marginalise out the function values over the prior defined by the kernel and \mathbf{X} . The marginal likelihood is

$$p(\mathbf{y} | \mathbf{X}) = \int p(\mathbf{y} | \mathbf{f}, \mathbf{X})p(\mathbf{f} | \mathbf{X}) d\mathbf{f} \quad (3.40)$$

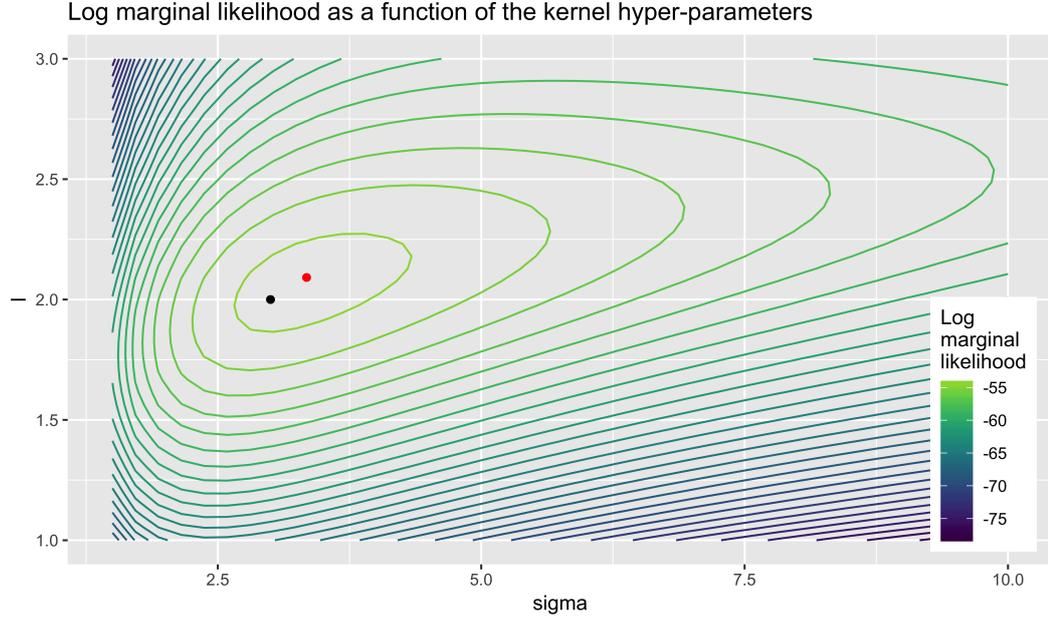


Figure 3.4: Contour plot of GP log marginal likelihood as a function of the kernel hyperparameters l and σ . The black dot shows the hyperparameters of the GP prior from which the data was sampled, and the red dot shows the hyperparameters found through optimizing the log marginal likelihood with respect to the hyperparameters using a gradient-based method.

where $\mathbf{f} | \mathbf{X} \sim \mathcal{N}(0, \mathbf{K})$ is derived from the GP prior, and the likelihood $\mathbf{y} | \mathbf{f}, \mathbf{X} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbf{I})$ is additive Gaussian noise. Since (per equation 3.17)

$$\mathbf{y} | \mathbf{X} \sim \mathcal{N}(0, \mathbf{K} + \sigma_n^2 \mathbf{I}) \quad (3.41)$$

$\mathbf{y} | \mathbf{X}$ has log marginal likelihood

$$\log p(\mathbf{y} | \mathbf{X}) = -\frac{1}{2} \left(\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y} + \log \det(\mathbf{K}_y) + n \log 2\pi \right) \quad (3.42)$$

where $\mathbf{K}_y = \mathbf{K} + \sigma_n^2 \mathbf{I}$. The three components of the marginal likelihood can be seen to encode respectively the data fit ($\mathbf{y}^T \mathbf{K}_y^{-1} \mathbf{y}$), a complexity penalty ($\log |\mathbf{K}_y|$) and a normalization term ($n \log 2\pi$).

We can now optimize the log marginal likelihood with respect to the kernel hyperparameters using gradient-based optimization methods. Setting $L = \log p(\mathbf{y} | \mathbf{X})$

and applying the equations in appendix D.1.3 we can determine:

$$\frac{\partial L}{\partial \theta_i} = \frac{1}{2} \left(\mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \mathbf{K}_y^{-1} \mathbf{y} - \det(\mathbf{K}_y)^{-1} \det(\mathbf{K}_y) \operatorname{tr} \left(\mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) \right) \quad (3.43)$$

$$= \frac{1}{2} \left(\mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \mathbf{K}_y^{-1} \mathbf{y} - \operatorname{tr} \left(\mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) \right) \quad (3.44)$$

This can be further simplified by considering that

$$\mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \mathbf{K}_y^{-1} \mathbf{y} = \operatorname{tr} \left(\mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \mathbf{K}_y^{-1} \mathbf{y} \right) \quad (3.45)$$

$$= \operatorname{tr} \left(\mathbf{K}_y^{-1} \mathbf{y} \mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) \quad (3.46)$$

since the l.h.s. is a scalar (and so equal to its trace), and the trace is invariant under cyclic permutations of products. Equation 3.44 then becomes:

$$\frac{\partial L}{\partial \theta_i} = \frac{1}{2} \left(\operatorname{tr} \left(\mathbf{K}_y^{-1} \mathbf{y} \mathbf{y}^T \mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) - \operatorname{tr} \left(\mathbf{K}_y^{-1} \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) \right) \quad (3.47)$$

$$= \frac{1}{2} \operatorname{tr} \left(\left(\mathbf{K}_y^{-1} \mathbf{y} \mathbf{y}^T \mathbf{K}_y^{-1} - \mathbf{K}_y^{-1} \right) \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) \quad (3.48)$$

$$= \frac{1}{2} \operatorname{tr} \left(\left(\boldsymbol{\alpha} \boldsymbol{\alpha}^T - \mathbf{K}_y^{-1} \right) \frac{\partial \mathbf{K}_y}{\partial \theta_i} \right) \quad (3.49)$$

where $\boldsymbol{\alpha} = \mathbf{K}_y^{-1} \mathbf{y}$. With this in hand, gradients with respect to the hyperparameters can be calculated efficiently, using the Cholesky decomposition⁶ of \mathbf{K}_y to calculate \mathbf{K}_y^{-1} , or equivalently (but with improved numerical stability) to solve $\mathbf{K}_y \mathbf{U} = \frac{\partial \mathbf{K}_y}{\partial \theta_i}$ for \mathbf{U} .

With $\frac{\partial L}{\partial \theta_i}$ in hand, we can now optimize L using a standard gradient-based optimizer. Our R implementation of GPLVM uses the limited-memory modification of the BFGS method of Byrd et al. [1995] as implemented in the `optimx` package [Nash, 2014]. Other methods have been used; Lawrence [2004] use scaled conjugate gradients.

⁶Given a Hermitian positive-definite matrix \mathbf{A} , there exists a unique lower triangular matrix \mathbf{L} such that $\mathbf{A} = \mathbf{L}\mathbf{L}^*$. This is known as the Cholesky decomposition.

3.2.5 PCA, probabilistic PCA, and dual probabilistic PCA

Principal component analysis (PCA)

PCA is a common dimensionality reduction technique. From a geometric perspective, given N samples of some d -dimensional observed data \mathbf{X} ,⁷ PCA seeks to find an orthogonal linear projection $\mathbf{Z} = \mathbf{W}^T \mathbf{X}$ which projects \mathbf{X} onto a q -dimensional subspace in such a way that the squared reconstruction error of \mathbf{X} from its projection, $\|\mathbf{W}\mathbf{Z} - \mathbf{X}\|^2$, is minimized [Pearson, 1901]. Another definition of PCA is that \mathbf{W}^T is the orthogonal linear projection which maximises the variance in the projected space, with the variance decreasing along the dimensions [Hotelling, 1933], that is, the variance is highest along the first dimension, second-highest along the second, etc.

It is from the latter definition that the standard derivation of PCA stems, by either finding the q largest eigenvalues and associated eigenvectors of the covariance matrix $\mathbf{X}^T \mathbf{X} / N$ and setting \mathbf{w}_i to be the i^{th} eigenvector, or by finding the singular value decomposition (SVD) of $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$, where $\mathbf{U}^* \mathbf{U} = \mathbf{V}^* \mathbf{V} = \mathbf{I}$ and $\mathbf{\Sigma}$ is a diagonal matrix with non-negative entries, and setting \mathbf{W} equal to the matrix derived from the first q columns of \mathbf{V} . The SVD method is more numerically stable and generally preferred in implementations of PCA.

Probabilistic PCA

Tipping and Bishop [1999] presented PPCA, in which PCA may be derived from a probabilistic model of the observed data by maximum likelihood estimation. If \mathbf{x} is our observed data with dimension d , \mathbf{z} is the latent (projected) representation of our data, σ^2 is the variance of additive isotropic Gaussian noise and \mathbf{W}^T the projection from observed to latent space, we model the conditional probability of \mathbf{x} given \mathbf{z} as

$$\mathbf{x} \mid \mathbf{z}, \mathbf{W}, \sigma \sim \mathcal{N}(\mathbf{W}\mathbf{z}, \sigma^2 \mathbf{I}) \quad (3.50)$$

Setting the marginal distribution over the latent variables \mathbf{z} to be $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and integrating out the latent variables gives

$$\mathbf{x} \mid \mathbf{W}, \sigma \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \quad (3.51)$$

⁷Here and elsewhere in this section it is assumed without loss of generality that \mathbf{X} has been centred, that is, the sample mean $\bar{\mathbf{x}} = \mathbf{0}$. If \mathbf{X} is not centred we can simply apply the transformation $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}$ to each sample and proceed as described.

where $\mathbf{K} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$. The corresponding log-likelihood for a sample of size N is

$$L = -\frac{N}{2} \left(d \log(2\pi) + \log|\mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}| + \text{tr} \left(\frac{1}{N} \mathbf{K}^{-1} \mathbf{X}^T \mathbf{X} \right) \right) \quad (3.52)$$

The likelihood is maximised when

$$\mathbf{W}_{ML} = \mathbf{U}_q \left(\Lambda_q - \sigma^2 \mathbf{I} \right)^{\frac{1}{2}} \mathbf{R} \quad (3.53)$$

where the q columns of \mathbf{U}_q are the q leading eigenvectors of $\frac{1}{N} \mathbf{X}^T \mathbf{X}$, with corresponding eigenvalues λ_j on the diagonal of the diagonal matrix Λ_q , and \mathbf{R} is an arbitrary orthogonal rotation matrix. At $\mathbf{W} = \mathbf{W}_{ML}$, the maximum likelihood estimator for σ^2 is

$$\sigma_{ML}^2 = \frac{1}{d-q} \sum_{j=q+1}^d \lambda_j \quad (3.54)$$

Dual probabilistic PCA

In PPCA, one marginalises equation 3.50 over the latent variables \mathbf{z} , and then optimizes the log marginal likelihood with respect to the loadings matrix \mathbf{W} to obtain a maximum likelihood estimate for \mathbf{W} . In dual PPCA, one marginalises over the loadings matrix \mathbf{W} to obtain a maximum likelihood estimate for \mathbf{z} [Lawrence, 2004]. We specify a prior distribution for \mathbf{W} as

$$w_{ij} \sim \mathcal{N}(0, \alpha^2) \quad (3.55)$$

and integrating over \mathbf{W} gives us a marginal likelihood for \mathbf{X}

$$p(\mathbf{X} | \mathbf{Z}, \sigma) = \frac{1}{(2\pi)^{dN/2} |\mathbf{K}|^{d/2}} \exp \left(-\frac{1}{2} \text{tr} \left(\mathbf{K}^{-1} \mathbf{X} \mathbf{X}^T \right) \right) \quad (3.56)$$

where $\mathbf{K} = \alpha^{-2} \mathbf{Z} \mathbf{Z}^T + \sigma^2 \mathbf{I}$, and the log marginal likelihood we seek to maximise is therefore

$$L = -\frac{1}{2} \left(dN \log(2\pi) + d \log|\mathbf{K}| + \text{tr} \left(\mathbf{K}^{-1} \mathbf{X} \mathbf{X}^T \right) \right) \quad (3.57)$$

Differentiating this with respect to \mathbf{Z} gives

$$\frac{\partial L}{\partial \mathbf{Z}} = \alpha^{-2} \mathbf{K}^{-1} \mathbf{X} \mathbf{X}^T \mathbf{K}^{-1} \mathbf{Z} - \alpha^{-2} d \mathbf{K}^{-1} \mathbf{Z} \quad (3.58)$$

and setting this equal to zero tells us that at the solution to the optimisation problem $\mathbf{X}\mathbf{X}^T\mathbf{K}^{-1}\mathbf{Z} = d\mathbf{Z}$. Substituting the SVD of $\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ into this gives:

$$\frac{1}{d}\mathbf{X}\mathbf{X}^T\left(\alpha^{-2}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T + \sigma^2\mathbf{I}\right)^{-1}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad (3.59)$$

$$\frac{1}{d}\mathbf{X}\mathbf{X}^T\left(\alpha^{-2}\mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T + \sigma^2\mathbf{I}\right)^{-1} = \mathbf{I} \quad (3.60)$$

$$\alpha^{-2}\mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T + \sigma^2\mathbf{I} = \frac{1}{d}\mathbf{X}\mathbf{X}^T \quad (3.61)$$

$$\mathbf{U}\left(\alpha^{-2}\mathbf{\Sigma}\mathbf{\Sigma}^T + \sigma^2\mathbf{I}\right)\mathbf{\Sigma} = \frac{1}{d}\mathbf{X}\mathbf{X}^T\mathbf{U}\mathbf{\Sigma} \quad (3.62)$$

and since $\mathbf{\Sigma}$ is diagonal, the columns of \mathbf{U} are eigenvectors of $\mathbf{X}\mathbf{X}^T/d$, and the diagonal entries of $\mathbf{\Sigma}$, Σ_{ii} , satisfy

$$\lambda_i = \alpha^{-2}\Sigma_{ii}^2 + \sigma^2 \quad (3.63)$$

where λ_i is the eigenvalue associated with the i^{th} column of \mathbf{U} . Hence the maximum likelihood estimate for \mathbf{Z} is

$$\mathbf{Z}_{ML} = \mathbf{U}_q \left(\frac{\alpha^2\Lambda_q}{d} - \sigma^2\alpha^2 \right)^{\frac{1}{2}} \mathbf{R} \quad (3.64)$$

where the q columns of \mathbf{U}_q are the q leading eigenvectors of $\mathbf{X}\mathbf{X}^T/d$, with corresponding eigenvalues λ_j on the diagonal of the diagonal matrix Λ_q , and \mathbf{R} is an arbitrary orthogonal rotation matrix. The solution can be shown to be equivalent to PCA [Tipping, 2001].

3.2.6 Gaussian process latent variable models

Examination of equation 3.56 reveals that the marginal distribution of $\mathbf{X} \mid \mathbf{Z}, \sigma$ is the product of d GPs with the linear kernel $K = \alpha^{-2}\mathbf{Z}\mathbf{Z}^T + \sigma^2\mathbf{I}$. A natural step is to investigate the effect of replacing the linear kernel with a non-linear kernel such as the squared exponential kernel. Doing so leads to the Gaussian process latent variable model (GPLVM) [Lawrence, 2004].

A GPLVM is a dimensionality reduction technique which assumes that the variables in your observed d -dimensional data (drawn from some space $\mathcal{X} = \mathbb{R}^d$) are a non-linear function of q unobserved latent variables, where the functions which define the mapping from the latent space \mathcal{Z} to \mathcal{X} are distributed as a GP over \mathcal{Z} . GPLVMs can be used as a feature extraction step in a data analysis pipeline [Andriluka et al.,

2008].

Given an observed space $\mathcal{X} = \mathbb{R}^d$, a latent space \mathcal{Z} , and a kernel function $k_{\mathcal{Z}}$ over \mathcal{Z} , the function $\mathbf{f}(\mathbf{z}) = (f_1(\mathbf{z}), \dots, f_d(\mathbf{z}))$, $f_i: \mathcal{Z} \rightarrow \mathbb{R}$ maps from \mathcal{Z} to the \mathcal{X} , and each f_i is distributed according to the GP

$$f_i \sim \mathcal{GP}(0, k_{\mathcal{Z}}) \quad i = 1, \dots, d \quad (3.65)$$

and for a design matrix \mathbf{X} of N samples drawn from \mathcal{X} with associated latent variables \mathbf{Z} , \mathbf{X} is distributed as

$$\mathbf{X}_{*j} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathcal{Z}} + \sigma_n^2 \mathbf{I}) \quad j = 1, \dots, d \quad (3.66)$$

where σ_n^2 is the variance of the additive Gaussian noise, $\mathbf{K}_{\mathcal{Z}}$ is the kernel matrix, whose (i, j) th element is $k(\mathbf{Z}_{i*}, \mathbf{Z}_{j*})$, and \mathbf{A}_{i*} denotes the i th row of \mathbf{A} . One can then optimize \mathbf{Z} , $\boldsymbol{\theta}$, and σ_n to maximize the log marginal likelihood of the model for a given data matrix \mathbf{X} .

3.3 Recent developments in Gaussian processes

Gaussian processes are a fertile field of research, with work progressing in a number of directions. In this section we review some notable recent developments in the literature relating to Gaussian process methods.

The Automatic Statistician Since 2015, Zoubin Ghahramani et al. have been developing a set of Gaussian process techniques under the Automatic Statistician Project, which “aims to automate data science, producing predictions and human-readable reports from raw datasets with minimal human intervention” [Steinruecken et al., 2019]. The Automatic Statistician attempts to automatically explore structure in data, progressively fitting kernels to the data using either Gaussian process regression or classification as appropriate. Kernels are selected using Bayesian model selection techniques. Unfortunately, this process can be exceedingly time consuming due to the model search operating in $\mathcal{O}(N^3)$ time, limiting its utility to small data sets. An exciting recent development in this area is presented in Kim and Teh [2018], which reduces the complexity of the algorithm to $\mathcal{O}(N^2)$ and so opens up the possibility of its practical use on much larger data sets.

Modelling large-scale spatial data Another active area of research is focused on reducing the computation complexity of modelling large-scale, heterogeneous spatial data. Nychka et al. [2015] proposes a multi-resolution approach, using basis functions positioned according to a rectangular grid in two dimensions, with each basis function being a scaled squared exponential kernel.⁸ Kernel coefficients are then modelled using a Markov random field organized by the grid locations of the kernels. More recently a number of methods are being developed, and an excellent summary of current developments in this field can be found in Heaton et al. [2018]. These include methods for the parallel computation of the posterior density [Paciorek et al., 2013], a multi-resolution approach which is compatible with distributed computing [Katzfuss, 2017], a divide-and-conquer approach which splits the data into subsets [Guhaniyogi and Banerjee, 2018], and a class of nearest-neighbour Gaussian process models which introduces sparsity into the problem based on nearest neighbours [Datta et al., 2016].

Fast GP approximations Although GPs are a popular modelling tool due to their flexibility and power, they suffer from very high computational costs which scale poorly with increasing amounts of data, both for optimizing the model and for drawing from the posterior. An ongoing theme of GP research is the effort to reduce this computational cost, and this is reflected in the focus of other articles we have covered in this section. Notable recent contributions include: Gramacy and Apley [2015] and Zhang et al. [2018], who propose improvements for the use of GPs for kriging by dynamically restricting the support of the Gaussian processes to local neighbourhoods of the points to be predicted, based on an active learning heuristics; Bostanabad et al. [2018] present a novel method of hyperparameter optimization which adjusts the noise of the model in a specific way to aid the optimization algorithm in finding optimal hyperparameters; Foreman-Mackey et al. [2017] present a novel GP method for modelling one-dimensional data which scales linearly with increasing data rather than cubically; and Ambikasaran et al. [2015] demonstrate that for many kernels the covariance matrix can be decomposed into the product of low-rank updates of the identity matrix, thereby reducing the complexity of model optimization to $\mathcal{O}(N(\log N)^2)$.

⁸Also known as a radial basis function (RBF) kernel, which is the term used in Nychka et al. [2015].

Chapter 4

Structured Gaussian Process Latent Variable Models

In this chapter, we present the structured Gaussian process latent variable model (SGPLVM), an extension of standard GPLVM to include a prior specifying a correlative structure between features in the observed data. Structure in the observed data is defined by an arbitrary covariance matrix over the features. SGPLVM is motivated by the desire to better model the FAIMS data seen in Chapter 2, but the model also has broader applicability. Examples of common structures this model can be used to represent include spatial structure, such as in images or geographical data, and temporal structure such as in time series data. However, the model is also applicable in a range of other circumstances, as seen below. We also present a number of experiments we have conducted to explore the properties of SGPLVM.

4.1 Structured GPLVM

4.1.1 Formulation of structured GPLVM

Current formulations of GPLVMs in the literature assume that for each variable in the observed data the mapping from the latent space to the data space is drawn independently from the GP over \mathbf{Z} . However, for some data this assumption is clearly incorrect. A notable and relevant example is FAIMS analysis, where the data for a sample can be viewed as an image (see Figure 2.1) and there is a high degree of correlation between neighbouring pixels in the image. GPLVM assumes that the values of neighbouring pixels are independent; this discards useful information which could be incorporated into the model to improve its output. In this section we present SGPLVM, which attempts to incorporate knowledge of structured correlations

between variables into the prior.

To move forwards we must formalize our notion of structured correlation between variables. We say data exhibits structured correlations between variables if each column i of \mathbf{X} can be associated with a point \mathbf{s}_i in a structure space $\mathcal{S} = \mathbb{R}^k$, with each dimension of \mathcal{S} corresponding to a different aspect of the structure in the data. We can then place an ARD kernel over \mathcal{S} , with inverse length-scales $\mathbf{l} = (l_1, l_2, \dots, l_k)$ and variance α , defined by

$$k_S(\mathbf{u}, \mathbf{v}) = \alpha \cdot \exp\left(-\sum_{i=1}^k l_i^2 \cdot (u_i - v_i)^2\right) \quad (4.1)$$

for $\mathbf{u}, \mathbf{v} \in \mathcal{S}$. This defines the prior covariance between variables. When optimized against the marginal log likelihood of the model, the ARD kernel can account for dimensions in the structure space in which there is not exploitable structure by setting the corresponding inverse length-scales close to zero.

Now, let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be our design matrix with n samples drawn from an observed space \mathcal{X} , where \mathcal{X} has structured correlation between variables. Let $\mathbf{Z} \in \mathbb{R}^{n \times q}$ be a matrix of latent variables drawn from a latent space \mathcal{Z} , with $q < p$. In a GPLVM we assume \mathbf{X} is sampled from a Gaussian process over functions from the latent space to the data space, and that therefore for the j^{th} column of \mathbf{X} , \mathbf{X}_{*j} , we have

$$\mathbf{X}_{*j} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_Z + \sigma^2 \mathbf{I}) \quad (4.2)$$

where \mathbf{K}_Z is a covariance matrix derived from a positive-definite covariance function applied to \mathbf{Z} .

From this viewpoint, each feature in the design matrix is modelled as an independent Gaussian process. However, by vectorizing the design matrix \mathbf{X} we can consider this as a single Gaussian process, with a block diagonal covariance matrix containing p copies of \mathbf{K}_Z :

$$\text{Vec}(\mathbf{X}) \sim \mathcal{N}\left(0, \begin{pmatrix} \left(\begin{matrix} \mathbf{K}_Z & 0 & \cdots & 0 \\ 0 & \mathbf{K}_Z & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \mathbf{K}_Z \end{matrix} \right) + \sigma^2 \mathbf{I} \end{pmatrix} \right) \quad (4.3)$$

or, written more compactly using the Kronecker product $\mathbf{I} \otimes \mathbf{K}_Z$ of \mathbf{I} and \mathbf{K}_Z :

$$\text{Vec}(\mathbf{X}) \sim \mathcal{N}\left(0, \mathbf{I} \otimes \mathbf{K}_Z + \sigma^2 \mathbf{I}\right)$$

Our goal is to introduce covariance between features into our prior. Just as the right-hand side of the Kronecker product is the prior covariance matrix between the rows of \mathbf{X} , the left-hand side of the Kronecker product can be interpreted as the prior covariance matrix between the columns of \mathbf{X} .¹ Replacing \mathbf{I} in the Kronecker product with a covariance matrix \mathbf{K}_S derived from the structure between features gives us the SGPLVM:

$$\text{vec}(\mathbf{X}) \sim \mathcal{N}\left(0, \mathbf{K}_S \otimes \mathbf{K}_Z + \sigma^2 \mathbf{I}\right) \quad (4.4)$$

This model has a log-likelihood of

$$L(\mathbf{Z}, \theta) = -\frac{1}{2} \left(\log|\mathbf{K}| + \log(2\pi) + \frac{1}{2} \text{vec}(\mathbf{X})^T \mathbf{K}^{-1} \text{vec}(\mathbf{X}) \right) \quad (4.5)$$

where $\mathbf{K} = \mathbf{K}_S \otimes \mathbf{K}_Z + \sigma^2 \mathbf{I}$, and θ is the vector of hyperparameters for the kernels \mathbf{K}_S and \mathbf{K}_Z . This can be optimized as a GP over $\mathcal{S} \times \mathcal{Z}$. As with GPLVM, we use the squared-exponential kernel

$$k(\mathbf{u}, \mathbf{v}) = \alpha \cdot \exp\left(-\sum_i l^2 \cdot (u_i - v_i)^2\right) \quad (4.6)$$

with variance α and inverse length-scale l as the default kernel over the latent space in our implementation.

4.1.2 Examples of structure spaces

Greyscale images If \mathbf{X}_1 consists of greyscale image data, then the i^{th} column of \mathbf{X}_1 will correspond to a pixel coördinate $\mathbf{s}_i = (x_i, y_i)$.

RGB-encoded images As a more complex example, if \mathbf{X}_2 consists of RGB image data, then the i^{th} column of \mathbf{X}_2 will correspond to a pixel coördinate (x_i, y_i) and a colour channel (red, green, or blue). One representation of this structure is $\mathbf{s}_i = (x_i, y_i, r_i, g_i, b_i)$, using a 1-of- k form for the colour channel, where $r_i = 1$ if

¹To see this, consider the alternative formulation of equation 4.3 as $\text{vec}(\mathbf{X}^T) \sim \mathcal{N}(0, \mathbf{K}_Z \otimes \mathbf{I} + \sigma^2 \mathbf{I})$ and note that \mathbf{K}_Z in this formulation is the covariance of the columns of \mathbf{X}^T .

column i represents a pixel in the red channel, and 0 otherwise. Similarly for g_i (green channel) and b_i (blue channel).

Time series data A non-image example, the structure space for time series data consists of the one-dimensional space consisting of the time at which measurements were taken.

Geographic data As a final example consider geographic data, such as air pollution observations taken at a number of weather stations dispersed around a country, stored in a matrix \mathbf{X}_4 . If observations are taken at regular intervals (daily, for example) then each row of \mathbf{X}_4 contains all the data observed at a given point in time, and each column of \mathbf{X}_4 contains data for a different weather station. The latitude and longitude of the weather stations can then be used to define the columns' locations in structure space.

Learning the dimensionality of the latent space

A common problem when attempting to produce a low-dimensional representation of data is establishing an appropriate number of dimensions k for the low-dimensional representation. Taking PCA as an example, there are a number of ways in which this problem has been approached: by the “scree test” [Cattell, 1966] (also known as the elbow method), where the eigenvalues associated with the principal components are plotted in order of magnitude in a scree plot, and the number of retained dimensions is selected to be where this plot exhibits an inflection point, or more informally an elbow; as a model selection problem, where each value of k is a different model of the data and the evidence for each is estimated using a Laplace approximation to the posterior [Minka, 2001] or the related Bayesian Information Criterion; by parallel analysis [Horn, 1965], where a randomised version of the dataset is analysed and principal components are retained when their eigenvalues are greater than the corresponding eigenvalues derived from the randomised data; or by treating the dimensionality of the representation as intrinsic to the model and selecting the effective latent dimension by maximum likelihood, as in Bayesian PCA [Bishop, 1999].

In SGPLVM, it is possible to place an ARD kernel (see equation 4.1) over the latent space to learn the effective dimension of the latent space as part of the model. \mathbf{Z} is randomly initialised with some number of dimensions which is larger than the expected latent dimensionality, and the ARD kernel will only retain the

necessary number of latent dimensions, setting the other length-scales close to zero. In addition, a Laplace prior can be placed over the length-scales to induce sparsity in the optimized values [Williams, 1995].

4.2 Stochastic optimization

4.2.1 Intractability and wild geese

Maximizing the log-likelihood of equation 4.4 requires the inversion of the covariance matrix $\mathbf{K}_S \otimes \mathbf{K}_Z + \sigma^2 \mathbf{I}$, an $nd \times nd$ matrix. For many data sets, even ones with a small number of samples, this is an intractably large matrix to invert. As an example, for a modestly-sized FAIMS analysis of 100 samples, $\mathbf{K}_S \otimes \mathbf{K}_Z + \sigma^2 \mathbf{I}$ is a $5,222,400 \times 5,222,400$ matrix.

Initially we attempted to solve this with a two-pronged approach: first, use a sparse kernel to represent the structure, so that \mathbf{K}_S was mostly zero; second, approximate the model in such a way that it was unnecessary to ever compute $\mathbf{K}_S \otimes \mathbf{K}_Z$ in its entirety.

For the first part, we used piecewise polynomial kernels with compact support [Wendland, 2004, ch. 9] to generate sparse covariance matrices. For the second part, to approximate equation 4.4 we attempted to optimize against the approximation

$$\begin{aligned} \text{vec}(\mathbf{X}) &\sim \mathcal{N}\left(0, \widetilde{\mathbf{K}}_S \otimes \widetilde{\mathbf{K}}_Z\right) \\ \widetilde{\mathbf{K}}_S &= \mathbf{K}_S + \sigma_S^2 \mathbf{I} \\ \widetilde{\mathbf{K}}_Z &= \mathbf{K}_Z + \sigma_Z^2 \mathbf{I} \end{aligned} \tag{4.7}$$

This is the Matrix Normal distribution, which has the following probability density function:

$$p(\mathbf{X} \mid 0, \widetilde{\mathbf{K}}_Z, \widetilde{\mathbf{K}}_S) = \frac{\exp\left(-\frac{1}{2} \text{tr}\left(\widetilde{\mathbf{K}}_S^{-1} \mathbf{X}^T \widetilde{\mathbf{K}}_Z^{-1} \mathbf{X}\right)\right)}{(2\pi)^{\frac{nd}{2}} \left|\widetilde{\mathbf{K}}_S\right|^{\frac{n}{2}} \left|\widetilde{\mathbf{K}}_Z\right|^{\frac{d}{2}}} \tag{4.8}$$

With this approximation it is only necessary to invert $\widetilde{\mathbf{K}}_S$ and $\widetilde{\mathbf{K}}_Z$ to calculate the likelihood, without needing to calculate the full covariance matrix of the GP.

Unfortunately, after implementing this in R it became apparent during testing that the approximation would frequently converge on a solution where $\mathbf{K}_Z \approx \mathbf{0}$, and $\sigma_Z^2 \approx 1$. This effectively turned off the contribution of \mathbf{K}_Z to the likelihood of the

model, and led the model to converge to poor latent representations of the data. An alternative approach was required.

4.2.2 Stochastic gradient descent

In its most basic form, gradient descent is an optimization algorithm which attempts to minimize some objective function $L(\mathbf{w})$ over \mathbf{w} by iteratively updating the current solution \mathbf{w}_i using the gradient of L , such that

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \gamma \nabla_{\mathbf{w}} L(\mathbf{w}_i)$$

for some learning rate $\gamma > 0$ and proposed solution \mathbf{w}_i .

SGD is a popular and effective algorithm for learning parameters in large-scale machine learning problems [Bottou, 2010]. In contrast to gradient descent, at each iteration one stochastically estimates the gradient of the objective function, rather than calculating the gradient exactly. We have developed an algorithm which applies the principles of stochastic gradient descent to optimizing the log-likelihood of SGPLVM as given in equation 4.5.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$; kernels $\mathbf{K}_Z, \mathbf{K}_S$ with hyperparameter vector $\boldsymbol{\theta}$ (to be optimized); proposed latent variables \mathbf{Z}_i ; and proposed hyperparameters $\boldsymbol{\theta}_i$, our update step to find $\mathbf{Z}_{i+1}, \boldsymbol{\theta}_{i+1}$ is as follows. First, select an active set \mathcal{I} of indices from $1 \dots n \cdot d$ of a fixed size m . We will now approximate ∇L by considering

$$\text{vec}(\mathbf{X})_{\mathcal{I}} \sim \mathcal{N}\left(0, [\mathbf{K}_S \otimes \mathbf{K}_Z + \sigma^2 \mathbf{I}]_{\mathcal{I}, \mathcal{I}}\right) \quad (4.9)$$

where $[\mathbf{A}]_{\mathcal{I}, \mathcal{I}}$ is the sub-matrix of \mathbf{A} formed from the rows and columns with indices in \mathcal{I} . The covariance matrix for this GP is $m \times m$, and so is computationally feasible to invert in reasonable time on a modern workstation for values of m up to around 10,000. We can now compute the gradient of the log-likelihood $L_{\mathcal{I}}$ of equation 4.9, $\nabla L_{\mathcal{I}}$. For standard stochastic gradient descent, we set

$$\mathbf{Z}_{i+1} = \mathbf{Z}_i + \gamma \nabla_{\mathbf{Z}} L_{\mathcal{I}}(\mathbf{Z}_i, \boldsymbol{\theta}_i) \quad (4.10)$$

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \gamma \nabla_{\boldsymbol{\theta}} L_{\mathcal{I}}(\mathbf{Z}_i, \boldsymbol{\theta}_i) \quad (4.11)$$

4.2.3 Problems with optimizing SGPLVM using SGD

SGD as described in Section 4.2.2 suffers from a number of drawbacks which cause problems when used to optimize an SGPLVM. We describe here a number of the issues we have observed.

Selecting the learning rate Selecting the learning rate γ presents a trade off between convergence time and stability. The smaller the step size, the more iterations are required to converge to a given optimum from the same starting point, and the less chance the algorithm will have to explore parameter space due to its more conservative movements. On the other hand, if the step size is too large, the algorithm may overshoot and oscillate around the optimum [Qian, 1999]. Too large a step size can also cause parameters to diverge, irrecoverably moving the algorithm away from optimal solutions. This latter point is a particular problem with SGPLVM. If the latent point $\mathbf{z} = (z_1, \dots, z_q)^T$ associated with one sample moves too far from the other latent points² then $k(\mathbf{z}, \tilde{\mathbf{z}}) \approx 0$ for all $\tilde{\mathbf{z}} \neq \mathbf{z}$, and $\frac{\partial L}{\partial z_i} \approx 0$, and the point is effectively lost in the wilderness for the rest of the optimization process unless either the length-scale increases significantly or other points are moved closer to it. This “ejection” of points from latent representations has been observed frequently when developing our SGPLVM R package, as well as the more prosaic severe divergence of kernel hyperparameters which again causes the algorithm to fail to converge to an optimum.

Differences in parameter scales The fixed learning rate of standard SGD also presents a problem when the parameters being optimised differ markedly in scale. An appropriate step size for one parameter may cause another parameter which operates on a larger scale to converge too slowly, or contrariwise may cause a parameter measured on a significantly smaller scale to diverge, causing convergence to fail entirely.

Sparse gradients Finally, the likelihood in approximation in equation 4.9 will generally have a sparse gradient with respect to the latent variables. First, if the size of the active set m is less than the number of samples then some samples will not be represented in the approximation and the gradient of the approximate likelihood with respect to the latent variables of the unrepresented points will be zero. However, even if every sample is represented in the active set, it is possible that a number

²Relative to the length-scale—with the squared exponential kernel, when two points are more than three length-scales apart their covariance rapidly diminishes to zero with increasing distance.

of the points in the active set will be too distant from any of the other points in the active set to have a meaningful effect on the gradient of the likelihood. For two points to be close enough to interact, they must be close both in the latent space \mathcal{Z} and in the structure space \mathcal{S} (relative to the length-scales of the associated kernels).

4.2.4 Improvements to SGD

There is a large body of literature attempting to improve on the basic SGD algorithm. In this section we describe four modifications of SGD which can provide improved performance. For simplicity of notation, in this section we will use $\boldsymbol{\theta}$ to represent the vector of all parameters to be optimized,³ rather than just the kernel hyperparameters.

Learning rate schedules

The most straightforward adaptation of SGD is the introduction of learning rate schedules [Darken et al., 1992], where the learning rate is some function of the iteration number, $\gamma = f(t)$, rather than being constant throughout. Usually this entails the learning rate decreasing over time, with the aim being to allow for large movements towards the optimum initially, then fine-tuning at lower learning rates to avoid the phenomenon of overshooting the optimum and oscillating around it. Unfortunately it does not address the main issue we have observed with SGD in the context of SGPLVM, which is rapid divergence of parameters into regions of parameter-space which are very distant from any optima.

Momentum

Another simple adaptation of SGD is the addition of “momentum” [Qian, 1999] to the optimization process, by analogy to physical momentum. If v_i is the update step for a given parameter θ in the i^{th} iteration, then v_{i+1} is calculated as

$$v_{i+1} = \eta v_i + \gamma \frac{\partial L}{\partial \theta}(\mathbf{Z}_i, \boldsymbol{\theta}_i) \quad (4.12)$$

where a fraction η of the previous update step is retained. This has the effect that repeated updates in the same direction accumulate, while oscillating updates cancel each other out, damping down oscillations. This also has the effect of adapting somewhat to the scale of the parameters—if one parameter has significantly further to go to reach an optimum value, the rate of change for that parameter accelerates

³Kernel hyperparameters and latent variables.

as the momentum builds. This also allows a slower learning rate to be used while still achieving convergence in reasonable time, avoiding parameter divergence.

Stochastic meta-descent (SMD)

Stochastic meta descent (SMD) [Bray et al., 2004a,b] attempts to improve the stability and convergence speed of SGD by adapting the step size independently for each parameter, taking into account the past history of step size effects to reduce sudden large changes in step size.

Given an initial step size vector \mathbf{a}_0 , a meta-step size μ , a decay rate of the step size effect average λ , and initialising the exponential average of the effect of past step sizes on the new parameter values, \mathbf{v}_0 , at zero, the update step is as follows:

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1}) \quad (4.13)$$

$$\mathbf{v}_t = \lambda \mathbf{v}_{t-1} + \mathbf{a}_{t-1} \cdot (\mathbf{g}_t - \lambda \mathbf{H}_t \mathbf{v}_{t-1}) \quad (4.14)$$

$$\mathbf{a}_t = \mathbf{a}_{t-1} \cdot \max\left(\frac{1}{2}, 1 + \mu \mathbf{v}_t \cdot \mathbf{g}_t\right) \quad (4.15)$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{a}_t \cdot \mathbf{g}_t \quad (4.16)$$

where $\mathbf{a} \cdot \mathbf{b}$ is the element-wise Hadamard product of vectors \mathbf{a} and \mathbf{b} , and \mathbf{H}_t is the Hessian of L at iteration t . Pearlmutter [1994] gives a method for quickly calculating an approximate value for the $\mathbf{H}_t \mathbf{v}_{t-1}$ term in equation 4.14. First we expand $\nabla_{\boldsymbol{\theta}} L$ around $\boldsymbol{\theta}_{t-1}$:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1}) + \mathbf{H}_t \Delta\boldsymbol{\theta} + \mathcal{O}(\|\Delta\boldsymbol{\theta}\|^2) \quad (4.17)$$

Now set $\Delta\boldsymbol{\theta} = r \mathbf{v}_{t-1}$ and re-arrange to obtain:

$$\mathbf{H}_t \mathbf{v}_{t-1} = \frac{\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1} + r \mathbf{v}_{t-1}) - \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1})}{r} + \mathcal{O}(r) \quad (4.18)$$

and taking the limit as $r \rightarrow 0$

$$\mathbf{H}_t \mathbf{v}_{t-1} = \left. \frac{\partial}{\partial r} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1} + r \mathbf{v}_{t-1}) \right|_{r=0} \quad (4.19)$$

which can be approximated using standard numerical differentiation techniques.

Adam

Adam⁴ is “a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement” [Kingma and Ba, 2014]. It maintains running estimates of the first and second moments of the gradient of the objective function with respect to each parameter, and uses these to adapt the learning rate for each parameter. Adam has a number of features which are particularly attractive when optimizing SGPLVM likelihood:

1. The magnitude of parameter updates are invariant to rescaling of the gradient. Changing the approximation size m for a given problem will rescale the likelihood, so scale invariance allows approximation size to be tuned independently of the gradient descent hyperparameters.
2. Step sizes are bounded by either the step size hyperparameter or some constant multiple of it, making it less likely that a parameter diverges due to an abnormally large gradient approximation.
3. The fact that the update steps are derived from exponentially-decaying averages of the first and second moments of the gradient means Adam is effective at optimizing problems with sparse gradients such as SGPLVM

The Adam update step is calculated as follows. Given a step size α , moment decay rates $\beta_1, \beta_2 \in [0, 1)$, we initialise the first and second moment estimates $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{v}_0 = \mathbf{0}$. Then at iteration t , we calculate the update step as follows:

$$\mathbf{g}_t = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1}) \quad (4.20)$$

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t \quad (4.21)$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2 \quad (4.22)$$

$$\widehat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \quad (4.23)$$

$$\widehat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \quad (4.24)$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \frac{\alpha \widehat{\mathbf{m}}_t}{\sqrt{\widehat{\mathbf{v}}_t + \epsilon}} \quad (4.25)$$

where vector arithmetic is performed elementwise. Equations 4.21 and 4.22 update the estimates for the first and second moments respectively. Initialising the moments at zero leads to bias in the estimates given by these updates, so equations 4.23 and

⁴The name Adam is derived from “adaptive moment estimation”.

4.24 perform bias-correction on the moments. Equation 4.25 updates the estimate of the optimum parameters, with the Adam update rule adapting the step size to the current state of each parameter.

4.2.5 Introducing back-constraints

Unlike many dimensionality-reduction techniques (e.g. PCA, multidimensional scaling (MDS) [Kruskal, 1964], and ISOMAP [Tenenbaum et al., 2000]) which preserve local distances in observed data when generating a latent representation, GPLVMs and by extension SGPLVMs produce dissimilarity-preserving latent space representations. Because GPLVMs provide a smooth mapping from the latent space to the data space, points which are dissimilar in the observed data must also be dissimilar in the latent representation. However, no equivalent guarantee can be made for points which are close together in the observed data being close together in the latent space. Put another way, a smooth mapping such as that induced by a GPLVM may take the same value for two distant inputs (the input in this case being the latent space, while the “output” value is the observed space), but cannot take wildly different values for two similar inputs. This can lead to fragmented latent representations.

Lawrence and Quiñonero-Candela [2006] introduced back-constraints for GPLVMs to preserve locality in the latent space, where one constrains the latent variables \mathbf{Z} to be the result of a smooth mapping from the observed data. This ensures that points that are close together in the data space will also be close together in the latent space. We extended the same approach to SGPLVMs. Setting $\mathbf{Z} = f(\mathbf{X}; \mathbf{A})$ for some appropriate smooth function f , one uses the chain rule and $\frac{\partial \mathbf{Z}}{\partial \mathbf{A}_{ij}}$ to optimize over \mathbf{A} rather than \mathbf{Z} , learning a mapping from the data to the latent space rather than learning the latent representation directly. To achieve this, we constrain \mathbf{Z} to be generated by a squared exponential kernel mapping from \mathbf{X} , with length-scale l_{BC} and parameters \mathbf{A} . We then optimize the marginal likelihood of the SGPLVM model with respect to \mathbf{A} rather than \mathbf{Z} , with the value of l_{BC} set per dataset by experimentation.

Specifically, given data \mathbf{X} , function parameters \mathbf{A} , and back-constraint length-scale l_{BC} , we calculate a covariance matrix \mathbf{K} as

$$K_{ij} = \exp\left(-\frac{\|\mathbf{X}_{i*} - \mathbf{X}_{j*}\|^2}{2l_{BC}^2}\right) \quad (4.26)$$

and then set

$$\mathbf{Z} = \mathbf{K} \mathbf{A} \tag{4.27}$$

Setting the back-constraint length-scale

It is necessary to set the length-scale of the back-constraint l_{BC} for each optimization problem, with the appropriate length-scale depending on the data. Too short a length-scale will be equivalent to GPLVM without back-constraints, since no two points will be close enough to interact and $\mathbf{A} \approx \mathbf{Z}$. Setting l_{BC} too high, on the other hand, will provide too tight a constraint on the latent space, and will lead to the back-constraint mapping dominating the latent representation found.

To automate this process, we developed a heuristic to identify an appropriate length-scale for a given dataset, which gives good performance for a range of problems in practice. Given $p, c \in (0, 1)$ we compute l_{BC} such that $k(\mathbf{x}_i, \mathbf{x}_j) < c$ in 100 $p\%$ of cases. This is straightforward to calculate using the distance matrix \mathbf{D} of \mathbf{X} .⁵ Noting that

$$K_{ij} = \exp\left(-\frac{D_{ij}^2}{2l_{\text{BC}}^2}\right) \tag{4.28}$$

we then set l_{BC} to be the 100 p^{th} centile of

$$\left(-D_{ij}\sqrt{2}\log(c)\right)_{i \neq j} \tag{4.29}$$

4.3 Experiments on artificial data

We sampled artificial datasets (Figure 4.1) from SGPLVM models with known parameters and latent spaces, and evaluated the ability of the SGPLVM training algorithm to recover both the latent space and the hyperparameters used to generate the data. One dataset was generated from a latent space with well separated clusters to qualitatively assess the latent space reconstruction. For quantitative assessment of latent space reconstruction, twenty datasets were generated from a latent space with overlapping clusters. The reconstruction error was assessed using Procrustes analysis [Gower, 1975], where the error was defined as the minimum root-mean-square error (RMSE) between the true latent space and all conformal affine transformations⁶ of the reconstructed latent space.

⁵The distance matrix of \mathbf{X} is the matrix \mathbf{D} such that $D_{ij} = \|\mathbf{X}_{i*} - \mathbf{X}_{j*}\|$.

⁶That is, transformations consisting of some combination of translation, rotation, reflection, and uniform scaling.

Dataset	# of samples	Sample dimensions	Structure kernel length-scale	Structure kernel variance	Latent kernel length-scale	Latent kernel variance	Additive Gaussian noise variance
Close clusters	100	5x5	2	1	2	1	0.01
Distant clusters	100	5x5	3	1	0.4	1	0.01
SGD convergence	200	10x10	3	1	2	1	0.01

Table 4.1: Generating parameters for the artificial datasets discussed in this chapter.

Copies of the latent and observed data for all datasets discussed in this chapter are available online in comma-separated values (CSV) form at https://www.mattdneal.com/thesis_data.shtml. Table 4.1 lists the generating parameters of the datasets. All datasets were generated using the `sample.from.model` function from our GPLVM R package.

4.4 Stochastic gradient descent variants

We tested the convergence properties for optimizing SGPLVM of the four SGD variants covered in section 4.2.3: standard SGD; SGD with momentum; Adam; and SMD. To assess convergence we generated 200 samples from an SGPLVM, shown in figure 4.2, with a 10x10 structure to each sample, length-scales of 3 and 2 for the structure and latent kernels respectively, and additive Gaussian noise with a variance of 0.1.

Figures 4.3 to 4.6 show convergence rates for the four algorithms over 10,000 iterations at different step sizes. Figure 4.7 compares the convergence of the best step size for each algorithm. Table 4.2 gives the convergence times and log likelihoods converged to.

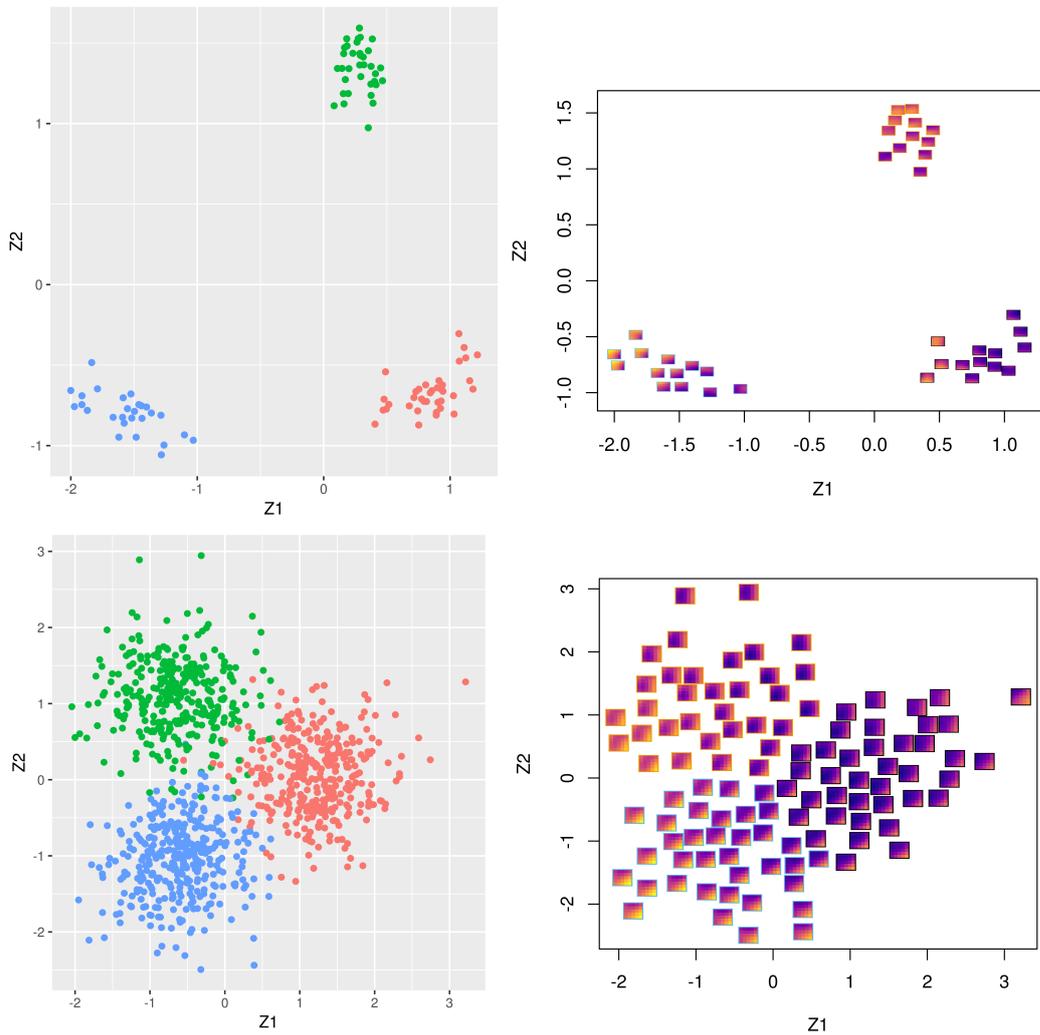


Figure 4.1: To generate the artificial data used in this chapter, we first generated random clustered 2-dimensional datasets to serve as the true latent spaces, one with distinct clusters (top row) and one with overlapping clusters (bottom row). We then sampled a 25-dimensional dataset with a 5x5 structure from an SGPLVM model over this latent space to serve as our observed data. Left: the underlying latent spaces used to generate our artificial data. Right: image representations of the data sampled from the SGPLVM models. Samples are plotted in a random order on the same axes as the underlying latent space, with their centroid at the coordinates of the associated latent point. Images are only plotted if they do not overlap with any already-plotted images. The colour bar for the right-hand plots is the same as that used for plotting FAIMS data in Chapter 2 (see for example fig. 2.20).

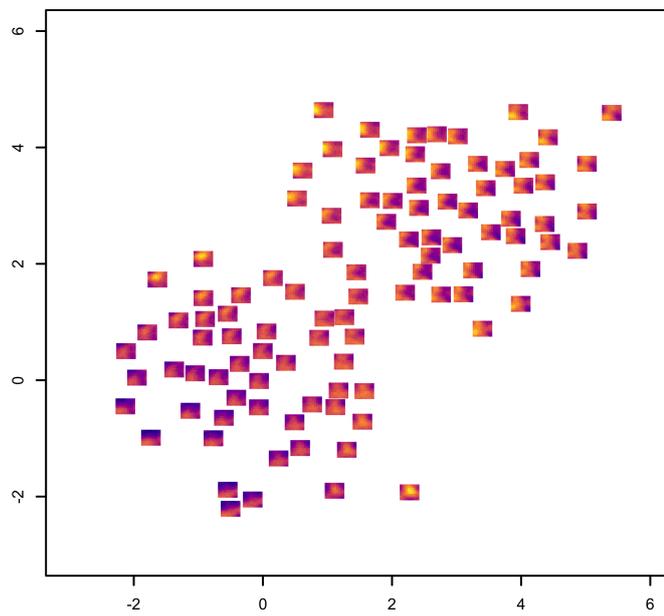


Figure 4.2: A plot of the artificial data used for assessing SGD algorithm convergence properties, using the same method for plotting as was used in Figure 4.1. 200 points were sampled from an SGPLVM with a 10×10 structure, with length-scales of 3 and 2 for the structure and latent kernels respectively, and additive Gaussian noise with a variance of 0.1.

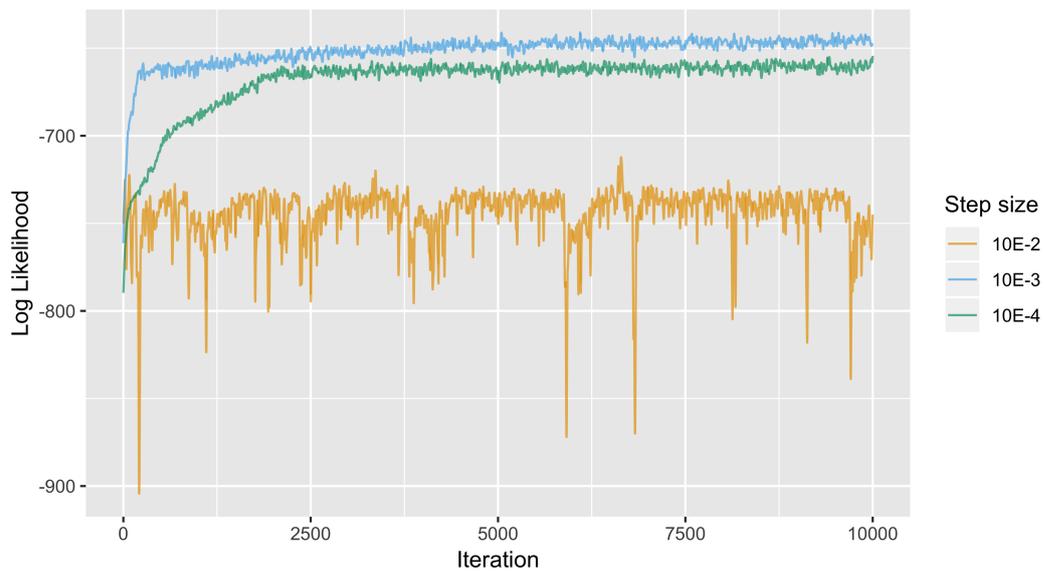


Figure 4.3: Likelihood convergence for standard SGD. Varying the step size has a marked impact on the ability of the algorithm to converge to an optimal solution.

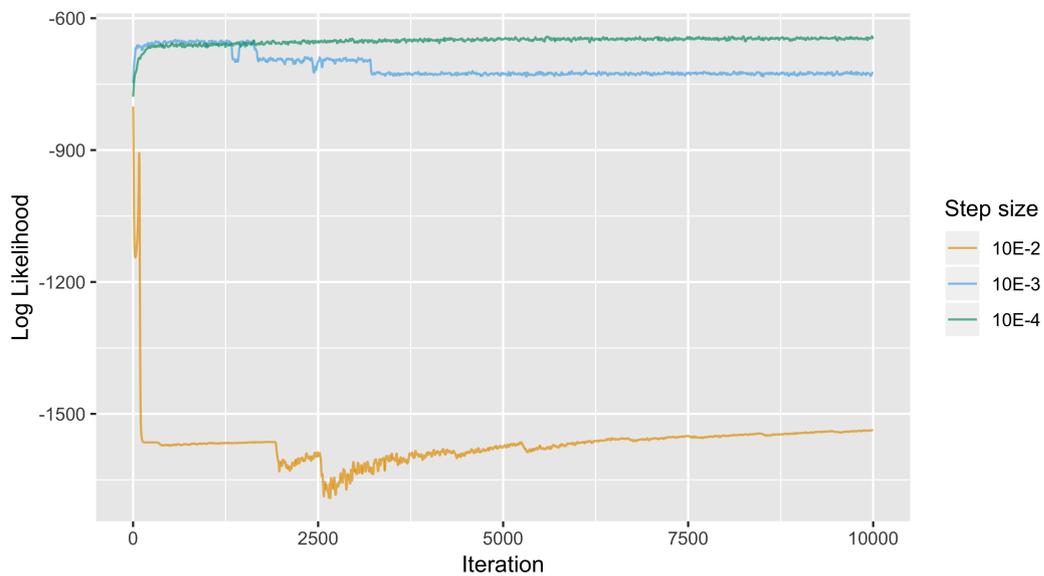


Figure 4.4: Likelihood convergence for SGD with momentum. As with standard SGD, varying the step size has a marked impact on the ability of the algorithm to converge to an optimal solution.

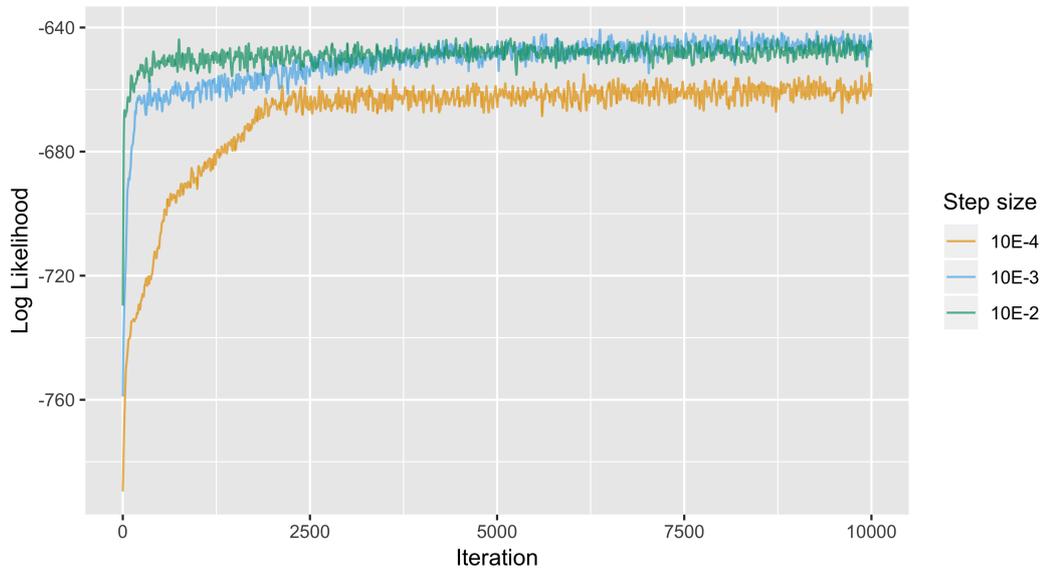


Figure 4.5: Likelihood convergence for SMD. SMD converges more quickly to an optimal solution than standard SGD, and is also more robust to the choice of step size.

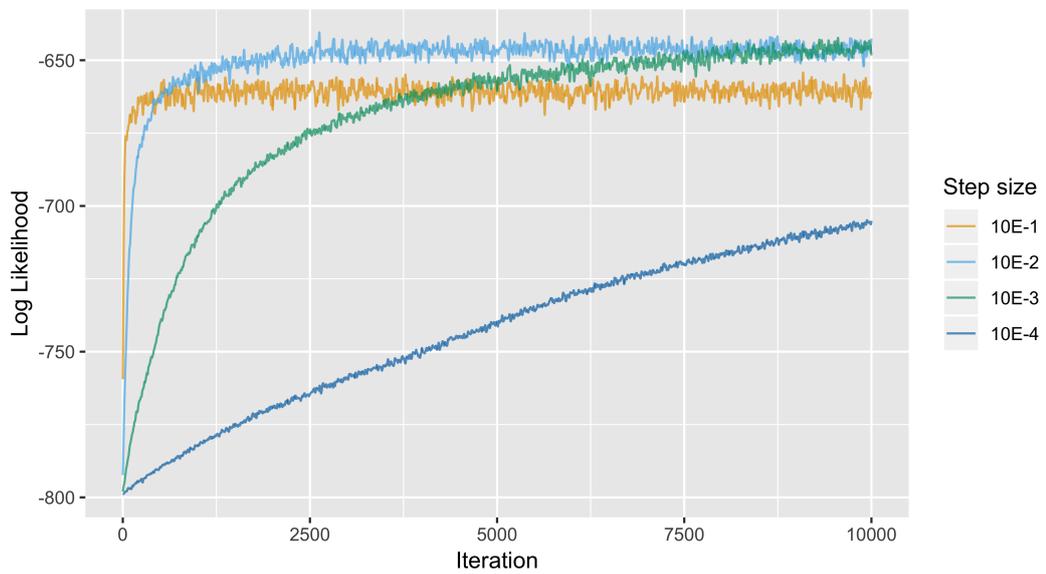


Figure 4.6: Likelihood convergence for Adam. Adam converges to an optimal solution at a step size an order of magnitude larger than the other SGD algorithms considered, and appears to converge robustly at a range of step sizes, although more slowly than SMD.

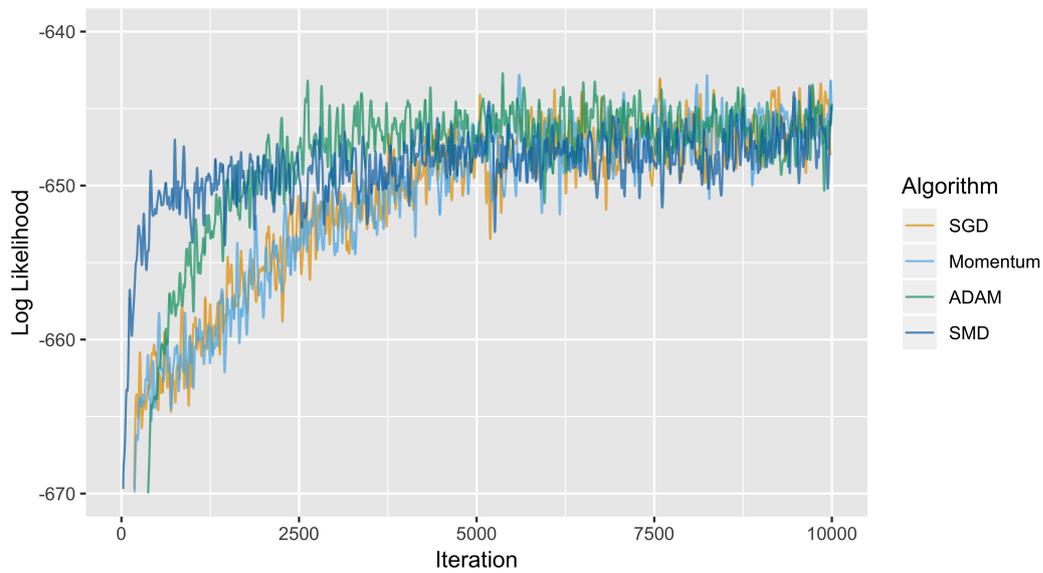


Figure 4.7: Comparison of likelihood convergence for different SGD algorithms. The best performing step size for each algorithm is shown here: 10^{-3} for SGD, 10^{-4} for momentum, 10^{-2} for Adam, and 10^{-2} for SMD. All algorithms ultimately converged to the same optimum. SGD and SGD with momentum perform comparably, and although SMD makes rapid gains initially, Adam ultimately achieves the optimum in the fewest iterations.

Algorithm	Step size (log ₁₀ scale)	Convergence time (iterations)	Converged log likelihood
SGD	-2	<i>Failed to converge</i>	<i>NA</i>
SGD	-3	5000	-646
SGD	-4	2500	-659
Momentum	-2	<i>Failed to converge</i>	<i>NA</i>
Momentum	-3	3250	-727
Momentum	-4	2500	-645
Adam	0	<i>Failed to converge</i>	<i>NA</i>
Adam	-1	1250	-661
Adam	-2	2500	-646
Adam	-3	8750	-645
Adam	-4	>10000	<i>NA</i>
SMD	-1	<i>Failed to converge</i>	<i>NA</i>
SMD	-2	1250	-647
SMD	-3	5000	-645
SMD	-4	2500	-660

Table 4.2: Convergence performance of the four SGD algorithms we assessed. All algorithms converged to the same optimum for at least one step size. SMD converged most quickly to the optimum log likelihood for most step sizes, however Adam converged reliably over the range of step sizes considered—SMD converged to less than the optimum log likelihood when the step size dropped to 10^{-4} , whereas Adam was still converging after 10,000 iterations, and identifying that the process has not yet converged is trivial whereas identifying that the process has converged to a suboptimal solution is not.

4.5 Comparison to GPLVM

4.5.1 Comparison of exact SGPLVM to a published GPLVM implementation

Our full implementation of SGPLVM with stochastic optimization (as described in Section 4.2.3) has been developed as an R [R Core Team, 2018a] package. After developing our R package, we extended the published Python package GPflow [Matthews et al., 2017] to include an implementation of SGPLVM optimized without approximation. This allows a comparison of SGPLVM to a published GPLVM implementation while minimizing differences due to implementation details or optimization methods. GPflow is a Python package which implements a number of Gaussian process models, including GPLVMs, using TensorFlow [Abadi et al., 2016], a popular open-source software library for machine intelligence.

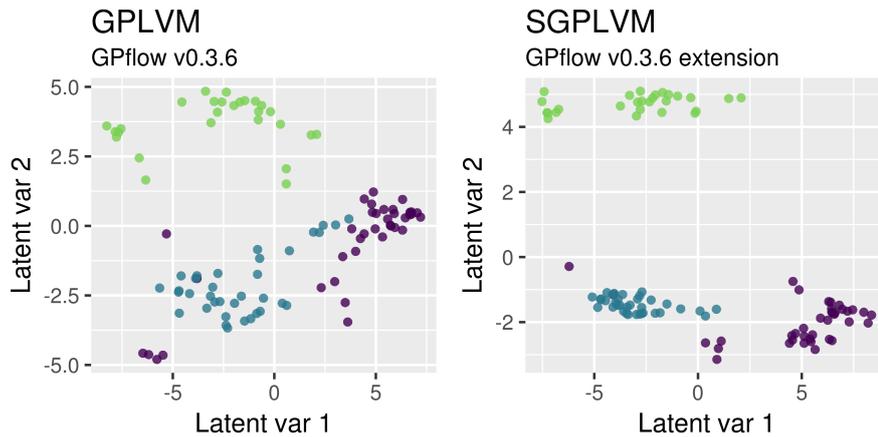


Figure 4.8: To compare SGPLVM with a published implementation of GPLVM, we implemented the SGPLVM model in the GPflow framework. Shown above is the result when a latent space is fitted for our distinct clusters artificial data using GPLVM (left) and SGPLVM (right) in GPflow. SGPLVM produces a clear improvement in cluster separation in the recovered latent space.

To ensure a fair comparison, we implemented the exact SGPLVM model without approximation or back-constraints as an extension to the GPflow package. The two implementations therefore differ only in the underlying mathematics, both being initialized in the same way and optimized through TensorFlow. The resulting latent spaces for GPLVM and SGPLVM are shown in Figure 4.8 for the distinct clusters dataset.

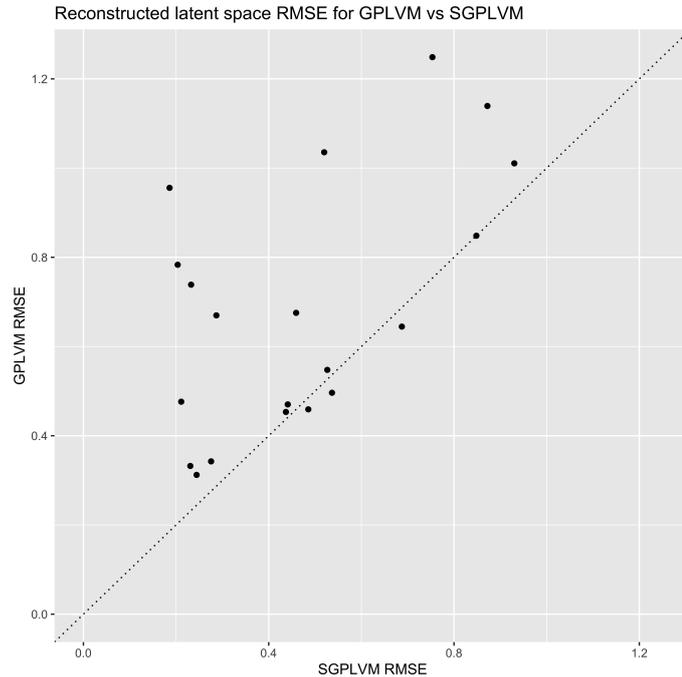


Figure 4.9: Reconstructed latent space RMSE for GPLVM versus SGPLVM for twenty datasets. Allowing a small margin of error SGPLVM performs at least as well as GPLVM in all datasets, and outperforms GPLVM in some datasets, reducing the RMSE by a factor of between two and five for six of the twenty datasets.

4.5.2 Comparison of approximate SGPLVM to GPLVM

For our twenty datasets with overlapping clusters, we compared the latent space reconstruction error of SGPLVM with GPLVM with back constraints [Lawrence and Quiñonero-Candela, 2006]. The results can be seen in Figure 4.9 and table 4.3. SGPLVM performed at least as well as GPLVM for all twenty data sets, and in a number of datasets shows a marked improvement in the recovered latent space.

Figure 4.10 shows the latent space recovered by SGPLVM with SGD for the distinct clusters dataset. Note that while the placement of clusters differs between the exact (Figure 4.8) and SGD versions, clusters are similarly well defined in both recovered latent spaces. Since SGPLVM (and GPLVMs more generally) are dissimilarity-preserving methods, the relative positions of distant points or clusters may be varied without significantly impacting the marginal likelihood of the latent variables. Importantly, the result from the approximate SGPLVM method does not differ markedly from the exact method seen in Figure 4.8.

<i>RMSE of latent representation</i>				
	PCA	GPLVM	SGPLVM	SGPLVM (ARD)
Dataset 1	0.99	0.85	0.85	0.88
Dataset 2	0.90	0.64	0.69	0.62
Dataset 3	1.07	1.14	0.87	0.82
Dataset 4	0.89	0.68	0.46	0.53
Dataset 5	0.72	0.48	0.21	0.58
Dataset 6	0.78	0.74	0.23	0.50
Dataset 7	0.74	0.45	0.44	0.54
Dataset 8	0.77	0.33	0.23	0.53
Dataset 9	0.95	0.50	0.54	0.64
Dataset 10	1.39	1.25	0.75	0.67
Dataset 11	0.48	0.46	0.49	0.46
Dataset 12	1.08	0.96	0.19	0.38
Dataset 13	0.98	1.01	0.93	0.64
Dataset 14	0.77	0.31	0.24	0.42
Dataset 15	0.98	1.04	0.52	0.66
Dataset 16	0.71	0.34	0.28	0.52
Dataset 17	1.04	0.55	0.53	0.94
Dataset 18	0.79	0.78	0.20	0.49
Dataset 19	1.23	0.47	0.44	0.61
Dataset 20	0.82	0.67	0.29	0.48
<i>Mean</i>	0.91	0.68	0.47	0.60
<i>S.D.</i>	0.20	0.28	0.24	0.15

Table 4.3: Comparison of performance of PCA, GPLVM, SGPLVM, and SGPLVM with automatic dimensionality selection using ARD. For each dataset the RMSE of the latent model compared to the true latent space is shown. All SGPLVM models were fitted using an approximation size of 512. The best result for each dataset is highlighted in bold.

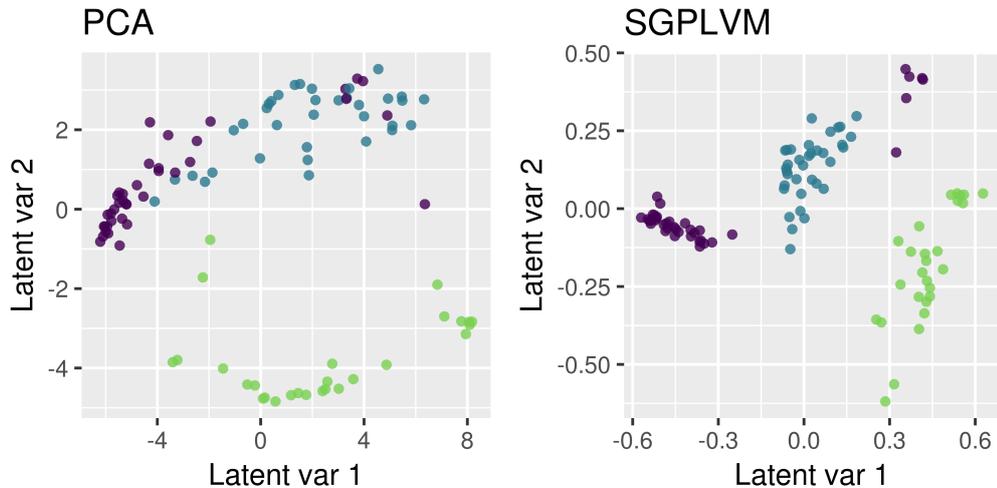


Figure 4.10: Latent space recovery on the distinct-clusters data using PCA (left) and SGPLVM optimized using stochastic gradient descent (right). SGPLVM again produces a clear improvement in cluster separation in the recovered latent space. It is also noteworthy that the approximate method produces a similar output to the exact method shown in Figure 4.8.

4.6 Impact of approximation size

To optimize the SGPLVM marginal likelihood directly it is necessary to compute the full SGPLVM kernel $\mathbf{K}_S \otimes \mathbf{K}_Z$. This precludes direct optimization from being used for anything other than modestly-sized data due to the size of the kernel matrix. To examine the model’s performance with higher-dimensional data and larger data sets, we implemented the approximate SGPLVM optimization algorithm described in Section 4.2.3 as an R package.

An important question is the impact of the approximation size on the recovered latent space. Figures 4.11 and 4.12 show the improvement over PCA of the recovered latent space RMSE for a range of approximation sizes over the twenty datasets generated from overlapping clusters, respectively using and not using ARD for automatic dimensionality determination. Table 4.5 and Table 4.4 show the data plotted in Figure 4.12 and Figure 4.11 respectively. RMSE was normalized using the RMSE of a 2-dimensional latent representation obtained through PCA to account for the differing difficulty of recovering an accurate latent space between the different datasets. An approximation size of 128 points was sufficient for most datasets to achieve their maximum accuracy, although accuracy still improved for some datasets

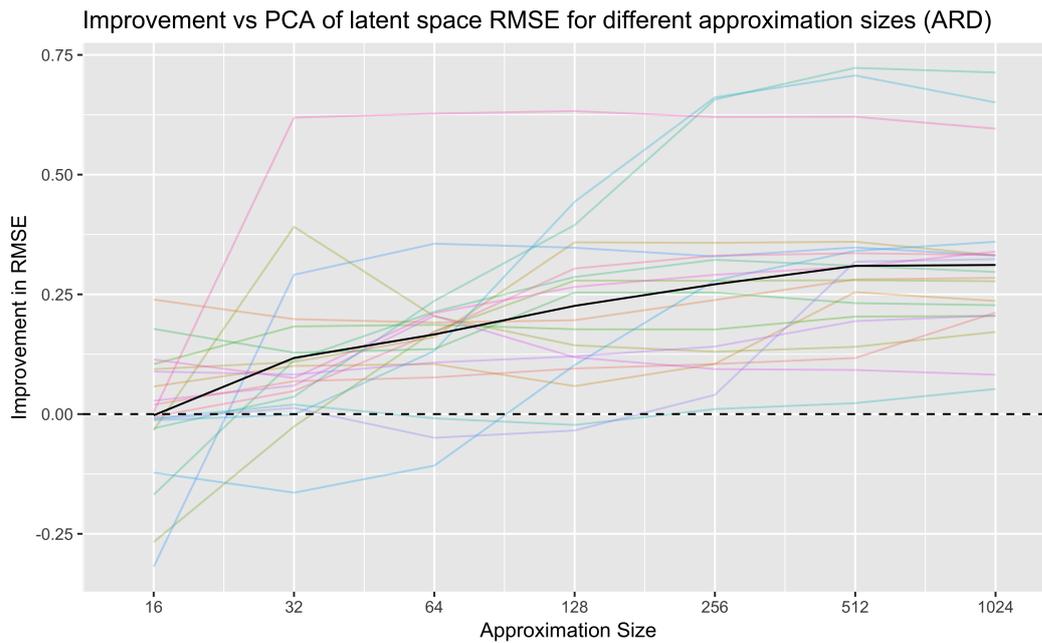


Figure 4.11: Improvement over PCA of the recovered latent space RMSE for a range of approximation sizes over the twenty datasets generated from overlapping clusters. The mean RMSE across all data sets for each approximation size is plotted in black, and trend lines for each dataset are plotted in colour. This figure shows RMSE for models recovered using ARD to automatically determine the appropriate latent space dimension. Overall performance improves up to an approximation size of 512 points, with no significant improvement seen when moving to an approximation size of 1024 points. On the basis of this all other analyses were performed with an approximation size of 512 points.

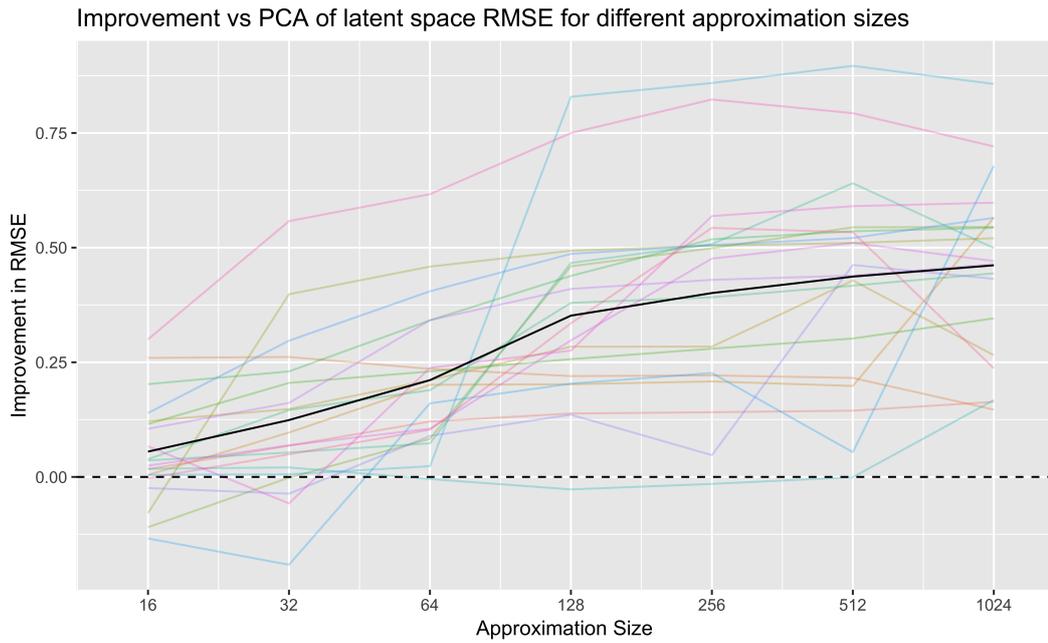


Figure 4.12: Improvement over PCA of the recovered latent space RMSE for a range of approximation sizes over the twenty datasets generated from overlapping clusters. The mean RMSE across all data sets for each approximation size is plotted in black, and trend lines for each dataset are plotted in colour. This figure shows RMSE for models recovered with a pre-specified latent dimension of 2. Overall performance improves up to an approximation size of 512 points, with no significant improvement seen when moving to an approximation size of 1024 points. On the basis of this all other analyses were performed with an approximation size of 512 points.

	<i>Approximation Size</i>						
	16	32	64	128	256	512	1024
Dataset 1	0.97	0.92	0.92	0.90	0.89	0.88	0.78
Dataset 2	0.66	0.70	0.71	0.71	0.67	0.62	0.62
Dataset 3	1.01	0.97	0.97	1.01	0.97	0.82	0.83
Dataset 4	0.79	0.78	0.73	0.53	0.53	0.53	0.56
Dataset 5	0.75	0.33	0.52	0.58	0.59	0.58	0.55
Dataset 6	1.04	0.80	0.60	0.50	0.50	0.50	0.50
Dataset 7	0.63	0.56	0.55	0.56	0.56	0.54	0.53
Dataset 8	0.59	0.64	0.63	0.51	0.51	0.53	0.54
Dataset 9	1.12	0.84	0.74	0.67	0.63	0.64	0.66
Dataset 10	1.42	1.36	1.16	1.00	0.74	0.67	0.68
Dataset 11	0.49	0.46	0.49	0.51	0.47	0.46	0.43
Dataset 12	1.10	1.08	0.95	0.64	0.42	0.38	0.43
Dataset 13	1.11	1.15	1.09	0.88	0.71	0.64	0.62
Dataset 14	1.08	0.47	0.41	0.42	0.44	0.42	0.43
Dataset 15	0.99	0.97	1.03	1.02	0.94	0.66	0.66
Dataset 16	0.63	0.63	0.61	0.59	0.57	0.52	0.51
Dataset 17	1.01	0.98	0.83	0.92	0.94	0.94	0.95
Dataset 18	0.68	0.72	0.58	0.53	0.50	0.49	0.45
Dataset 19	1.23	0.62	0.61	0.60	0.61	0.61	0.64
Dataset 20	0.83	0.77	0.65	0.52	0.49	0.48	0.49

Table 4.4: RMSE of the reconstructed latent space for a range of approximation sizes, when the dimension of the latent space is estimated automatically using an ARD kernel.

up to an approximation size of 512 points, with no discernible negative impact on those datasets which performed well at lower approximation sizes. SGPLVM outperformed PCA for all datasets at an approximation size of 256 points.

4.7 Partially observed data

Because of the correlation in the prior between features, it is possible to adapt the SGPLVM algorithm to handle partially unobserved data, with missing data inferred automatically both from other samples nearby in the latent space, and from other features nearby in the structure space. To investigate the ability of the algorithm to handle partially observed data we created partially observed versions of our artificial datasets with between 20% and 90% of the data randomly deleted, examples of which are shown in Figure 4.13.

	<i>Approximation Size</i>						
	16	32	64	128	256	512	1024
Dataset 1	0.98	0.92	0.87	0.85	0.85	0.85	0.83
Dataset 2	0.64	0.64	0.67	0.68	0.68	0.69	0.76
Dataset 3	1.07	0.97	0.87	0.87	0.86	0.87	0.51
Dataset 4	0.77	0.74	0.68	0.60	0.60	0.46	0.62
Dataset 5	0.80	0.32	0.26	0.23	0.22	0.21	0.20
Dataset 6	0.89	0.78	0.69	0.32	0.28	0.23	0.23
Dataset 7	0.62	0.53	0.51	0.48	0.46	0.44	0.39
Dataset 8	0.56	0.54	0.42	0.33	0.25	0.23	0.22
Dataset 9	0.91	0.81	0.76	0.57	0.56	0.54	0.51
Dataset 10	1.36	1.34	1.32	0.93	0.89	0.75	0.89
Dataset 11	0.47	0.46	0.49	0.51	0.50	0.49	0.32
Dataset 12	1.08	1.08	1.06	0.25	0.22	0.19	0.23
Dataset 13	1.12	1.18	0.82	0.78	0.76	0.93	0.31
Dataset 14	0.63	0.47	0.36	0.28	0.26	0.24	0.20
Dataset 15	1.01	1.02	0.89	0.85	0.93	0.52	0.55
Dataset 16	0.61	0.55	0.37	0.30	0.28	0.28	0.25
Dataset 17	1.01	0.97	0.93	0.74	0.56	0.53	0.57
Dataset 18	0.73	0.85	0.56	0.52	0.23	0.20	0.20
Dataset 19	0.93	0.68	0.62	0.48	0.41	0.44	0.51
Dataset 20	0.82	0.77	0.72	0.49	0.28	0.29	0.58

Table 4.5: RMSE of the reconstructed latent space for a range of approximation sizes, when the latent space is specified in advance to be two-dimensional.

As with the fully observed data we assessed the ability of the algorithm to recover the latent space of the generating model. For the distinct-clusters artificial data, the recovered latent space showed good separation between clusters, even when up to 75% of the data was unobserved (Figure 4.14).

4.8 Experiments on FAIMS data

The structure space for FAIMS data has the following dimensions:

- Compensation voltage, with 512 values evenly spaced between -3 V and 3 V , inclusive.
- Dispersion field strength, with 52 values evenly spaced between 0% and 100% , inclusive.
- Run number, with integer values from 1 to the number of runs per sample.

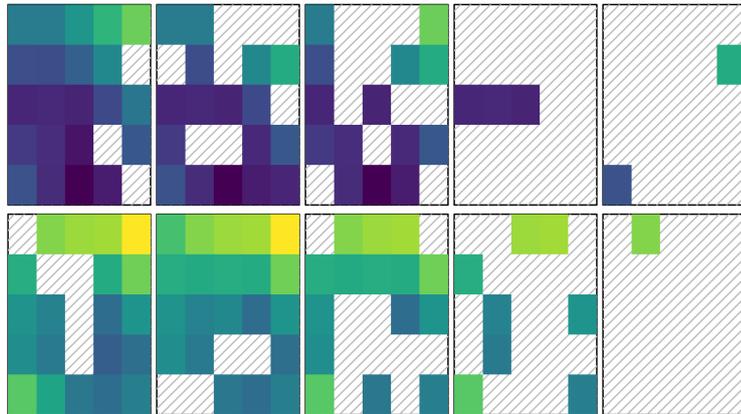


Figure 4.13: Examples of partially observed artificial data. From left to right: 10%, 20%, 50%, 70%, & 90% of data missing completely at random for two samples. Missing data are shown as grey hashed pixels.

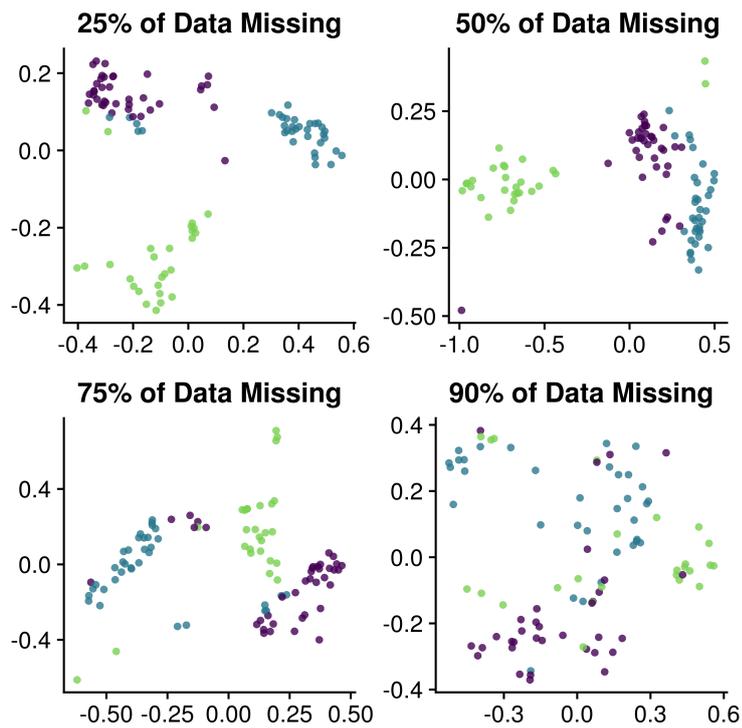


Figure 4.14: The recovered latent space for partially observed data with distinct clusters with 25%, 50%, 75%, & 90% of data missing. SGPLVM is dissimilarity-preserving, which is why cluster placement varies between the recovered latent spaces, but effective latent space recovery with visible cluster separation is still possible even with 75% of data missing.

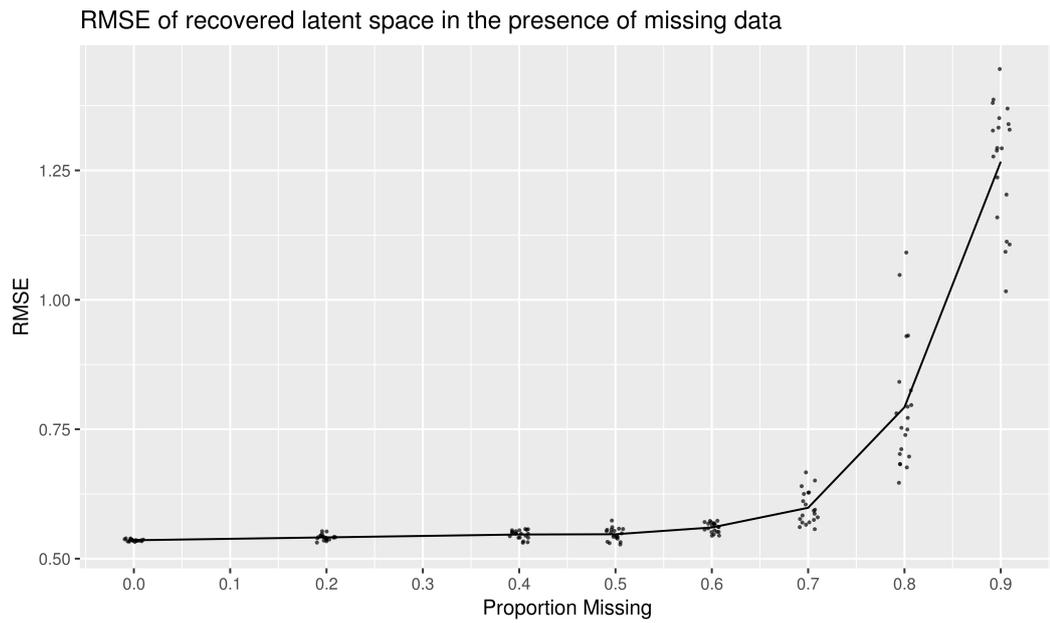


Figure 4.15: Mean SGPLVM-recovered latent space RMSE for twenty datasets generated from a single true latent space with overlapping clusters, plotted against the proportion of observed data missing. An ARD kernel was placed over the latent space to automatically determine the dimensionality of the latent space. The black line shows the mean RMSE, while RMSE for individual datasets are plotted as points (with jitter on the x-axis). RMSE of the recovered latent space only begins to be affected when between 60% and 70% of the data is missing.

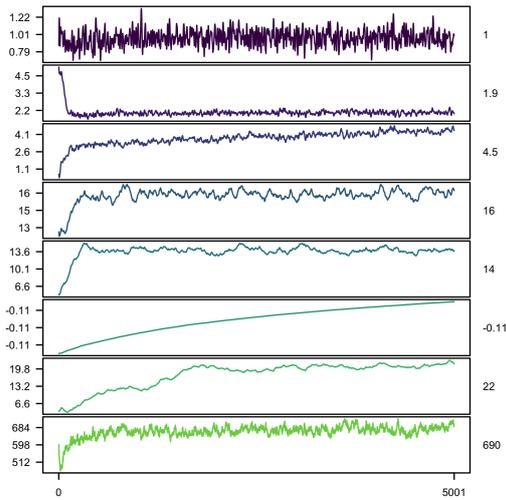
	Fraction of data missing:							
	0	0.2	0.4	0.5	0.6	0.7	0.8	0.9
Dataset 1	0.54	0.54	0.55	0.56	0.57	0.64	0.93	1.45
Dataset 2	0.53	0.54	0.54	0.54	0.56	0.57	0.70	1.34
Dataset 3	0.54	0.53	0.55	0.53	0.54	0.59	0.75	1.11
Dataset 4	0.54	0.54	0.56	0.54	0.56	0.63	0.75	1.39
Dataset 5	0.53	0.53	0.55	0.55	0.57	0.59	0.93	1.28
Dataset 6	0.53	0.54	0.55	0.55	0.54	0.58	0.68	1.37
Dataset 7	0.54	0.54	0.56	0.53	0.57	0.57	0.71	1.02
Dataset 8	0.53	0.55	0.56	0.55	0.56	0.57	0.68	1.33
Dataset 9	0.54	0.54	0.55	0.53	0.55	0.58	0.65	1.20
Dataset 10	0.54	0.55	0.55	0.56	0.57	0.56	0.68	1.24
Dataset 11	0.53	0.54	0.53	0.53	0.57	0.65	0.79	1.35
Dataset 12	0.54	0.54	0.54	0.56	0.55	0.63	0.74	1.29
Dataset 13	0.54	0.54	0.53	0.56	0.57	0.67	0.80	1.16
Dataset 14	0.54	0.54	0.55	0.55	0.55	0.63	1.09	1.33
Dataset 15	0.54	0.54	0.54	0.54	0.57	0.59	0.70	1.29
Dataset 16	0.53	0.54	0.53	0.55	0.55	0.61	0.83	1.38
Dataset 17	0.54	0.54	0.55	0.56	0.57	0.57	0.84	1.11
Dataset 18	0.53	0.54	0.55	0.54	0.56	0.56	0.77	1.33
Dataset 19	0.53	0.55	0.54	0.54	0.55	0.58	0.78	1.09
Dataset 20	0.53	0.54	0.54	0.57	0.56	0.60	1.05	1.29

Table 4.6: RMSE for recovery of the latent space in the presence of missing data for twenty datasets with varying fractions of the data missing completely at random. Latent spaces were fit using ARD to perform automatic dimensionality selection.

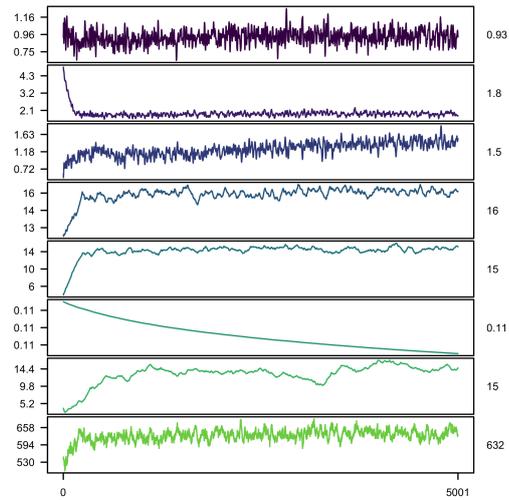
- Polarity, with values of -1 and 1 for negative and positive ion counts, respectively.

In our implementation, each point in structure space is assigned an integer value increasing from one, corresponding to the multi-dimensional array index associated with it in the R array data structure used by our `GPLVM` package, and as such all length-scales should be interpreted relative to this scaling, rather than the units given above.

We assessed the ability of SGPLVM to generate a two-dimensional visualisation of the FAIMS data from the Anastomotic Leakage studies discussed in sections 2.4.4 and 2.4.5 (hereafter referred to as AL Study 1 and AL Study 2, respectively), and compared it to PCA and GPLVM. To assess the recovered latent spaces we trained a Random Forest classifier on them to assess ability of the latent space to capture details of the data relevant to the classification problem. Table 4.7 gives the results for the Random Forest classifier trained on the latent spaces. Figure 4.17 shows the recovered latent spaces for the two data sets, and fig. 4.16 shows traces of the hyperparameter values for the SGPLVMs.



(a) AL Study 1



(b) AL Study 2

Figure 4.16: Hyperparameter and log likelihood traces for SGPLVMs trained on the two anastomotic leakage studies' FAIMS data. From top to bottom, the traces are: α , the variance of the SGPLVM; $\tau = 1/\sigma$, the precision of the additive Gaussian noise; the length-scale of the latent space kernel; the length-scale of the compensation voltage, the length-scale of the dispersion field strength; the length-scale of the polarity; the length-scale of the run number, and the log-likelihood. The numbers to the right of each trace give the final converged value.

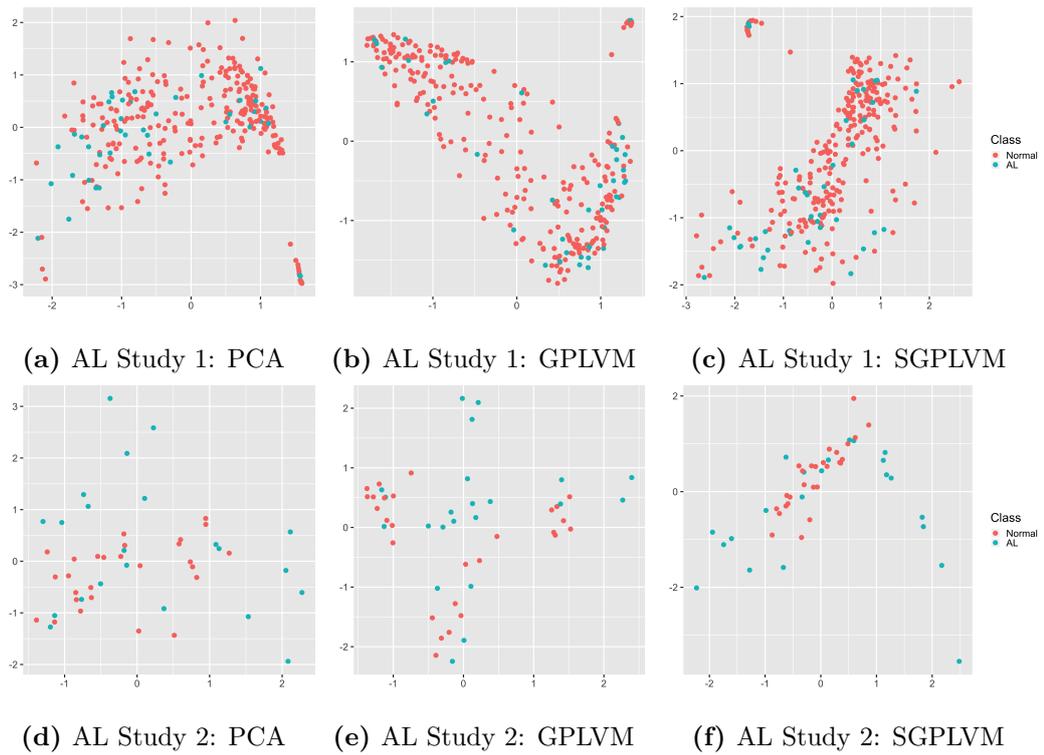


Figure 4.17: Recovered latent spaces for AL Studies 1 & 2 using PCA, GPLVM and SGPLVM. Normal samples are shown in red, and AL positive samples are shown in blue. None of the methods produces a latent representation with clear separation between the classes.

	<i>ROC AUC (95% CI)</i>			
	PCA	GPLVM	SGPLVM	SGPLVM - GPLVM
AL Study 1	0.59 (0.49, 0.70)	0.59 (0.49, 0.70)	0.70 (0.62, 0.79)	0.11 (0.020, 0.27)
AL Study 2	0.66 (0.48, 0.83)	0.65 (0.49, 0.82)	0.71 (0.55, 0.88)	0.062 (-0.074, 0.22)

Table 4.7: ROC AUCs for 2D latent spaces for FAIMS studies, generated using PCA, SGPLVM and GPLVM. Also shown is a bootstrapped 95 % confidence interval for the improvement of SGPLVM over GPLVM. For both studies SGPLVM was the only method to produce a latent space which allowed classification with an AUC better than chance at a 95 % confidence level. In the first study SGPLVM’s AUC was significantly better than GPLVM at a 95 % confidence level, however the difference was not statistically significant for the second study.

Chapter 5

SGPLVM and GP Regression: Implementation & Extensions

As part of developing the theory of SGPLVMs, we developed two substantial R packages and a number of smaller R packages. The two substantial packages are `gaussianProcess`, a package for Gaussian process regression which includes tools for generating complex compound kernels and performing model selection, and `GPLVM`, a package which fits a variety of GPLVM and SGPLVM models.¹ Both packages are available online at <https://github.com/mattdneal/gaussianProcess> and <https://github.com/mattdneal/GPLVM>. In this chapter we outline the functionality of these packages, and discuss the problems encountered during implementation and the solutions of the same.

5.1 The `gaussianProcess` package

The `gaussianProcess` package implements Gaussian process regression in R. Its main advantage over existing implementations is the use of a flexible, user-friendly method for generating and working with compound kernels, along with an extensive and extendable list of built-in kernels which are implemented in C for improved performance over native R code.

5.1.1 Kernel Model Trees

The core idea of the `gaussianProcess` package is the “model tree”, which represents compound kernels using a tree structure. This makes it easy for the user to specify a specific compound kernel using built in kernels and intuitive commands. Model trees

¹`gaussianProcess` and `GPLVM` consist of 6998 lines of code and 3777 lines of code respectively.

also enable the use of a model selection algorithm to explore the space of compound kernels. To do this, `gaussianProcess` performs a width-first search, attempting to add a kernel at a time to the current compound kernel and assessing model fit using a variety of metrics. Figure 5.1 shows an example model progression and the associated model trees.

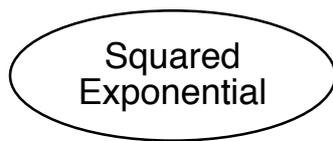
For a given model tree, it is straightforward to recursively compute the kernel matrix, along with the gradient and Hessian of the kernel, for optimization and assessing model fit.

5.1.2 Learning the optimum kernel

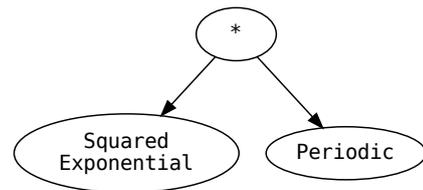
The following procedure is used to perform a search of possible compound kernels. First, a list of base kernels is defined, from which leaf nodes in the model tree will be taken. These may be standard kernels, such as a squared exponential kernel which is built in to the `gaussianProcess` package, or may be custom compound kernels defined as model trees themselves. Alongside this is a list of binary operations for combining kernels is defined, which will form the internal, non-leaf nodes of the model tree. Each of the basis kernels is then assessed for model fit, and the best basis kernel is placed at the root of a model tree.

After the n^{th} step we have a tree with $2n - 1$ nodes. To find the $n + 1^{\text{th}}$ model tree, we iterate over each node, combining it with each combination of binary operation and basis kernel. If there are b binary operations and k basis kernels, a total of $b \cdot k \cdot (2n - 1)$ new model trees will be generated and assessed for model fit. Optional, it is also possible to take backwards steps, assessing the impact of deleting each node in turn. If none of the model trees exceed the fit of the n^{th} model tree, the process stops and returns the current tree, otherwise the model tree with the highest fit is selected for the $n + 1^{\text{th}}$ step.

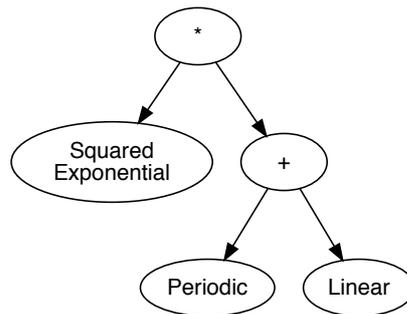
By way of example, we used this process to select an appropriate compound kernel for modelling the atmospheric CO₂ levels at the Mauna Loa observatory in Hawaii [Keeling and Whorf, 2018]. We split the data into a training set of 50 points selected at random, with the remaining 184 points making up a hold-out test set. Figure 5.2 shows the kernels selected at each stage, up to a maximum of three constituent basis kernels. Table 5.1 shows the BIC and RMSE for the test points for every compound kernel considered by the algorithm. Example code for generating this model based on the complete Mauna Loa data set can be seen in listing 5.1.



(a) Squared Exponential



(b) Squared Exponential * Periodic



(c) Squared Exponential * (Periodic + Linear)

Figure 5.1: Compound kernels can be represented as binary trees, with internal nodes representing operations such as addition and multiplication, and leaf nodes representing the constituent kernels. This figure shows a possible progression of the model search process, starting with (a) a squared exponential kernel. This is then multiplied by a periodic kernel in (b). The periodic kernel is then combined with a linear kernel in (c).

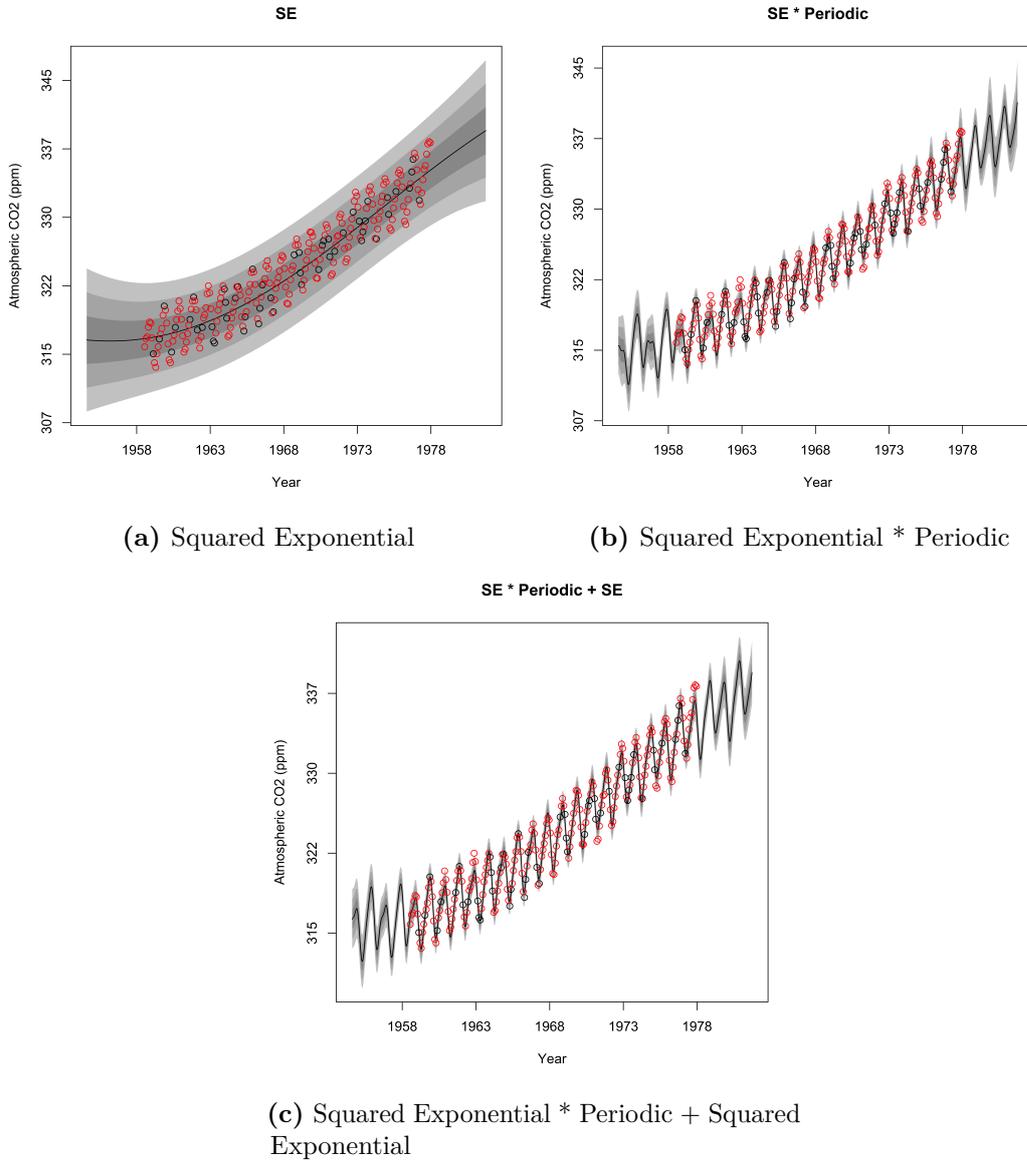


Figure 5.2: Automatically fitting compound kernels to the Mauna Loa atmospheric CO₂ concentration data. The kernels selected above were chosen entirely automatically using the algorithm described in the main text. Training points are shown in black, while test points are shown in red. The black line gives the posterior mean, while the shaded regions mark one, two, and three standard deviations from the mean. The algorithm first uses a squared exponential kernel (a) to capture the overall trend, before combining it with a period kernel (b) to capture the periodic variation around the trend line. In the final kernel (c), the addition of a second squared exponential kernel allows for local deviations from the global periodic pattern.

Kernel	BIC	Test set RMSE
squaredExponential.1	-44.9	1.86
neuralNetwork.1	-44.5	1.87
rationalQuadratic.1	-41	1.86
genPoly_2.1	-39.9	1.86
genPoly_1.1	-37.8	1.97
constant.1	58.9	5.55
periodic.1	59.5	5.97
(periodic.1 * squaredExponential.1)	-115	0.553
(periodic.1 + squaredExponential.1)	-88	0.671
(constant.1 + squaredExponential.1)	-41	1.86
(squaredExponential.1 * squaredExponential.2)	-41	1.86
(constant.1 * squaredExponential.1)	-41	1.86
(squaredExponential.1 + squaredExponential.2)	-40.7	1.86
(neuralNetwork.1 * squaredExponential.1)	-40.6	1.87
(genPoly_1.1 * squaredExponential.1)	-39.9	1.86
(genPoly_1.1 + squaredExponential.1)	-37.3	1.86
(genPoly_2.1 + squaredExponential.1)	-37.1	1.86
(rationalQuadratic.1 * squaredExponential.1)	-37.1	1.86
(genPoly_2.1 * squaredExponential.1)	-37.1	1.86
(rationalQuadratic.1 + squaredExponential.1)	-36.8	1.86
(neuralNetwork.1 + squaredExponential.1)	-36.3	1.86
((periodic.1 * squaredExponential.1) + squaredExponential.2)	-129	0.513
((neuralNetwork.1 + periodic.1) * squaredExponential.1)	-126	0.515
((periodic.1 * squaredExponential.1) + neuralNetwork.1)	-126	0.515
((periodic.1 + rationalQuadratic.1) * squaredExponential.1)	-125	0.513
((periodic.1 + periodic.2) * squaredExponential.1)	-125	0.516
((genPoly_2.1 + periodic.1) * squaredExponential.1)	-120	0.53
((periodic.1 + squaredExponential.2) * squaredExponential.1)	-116	0.541
((genPoly_1.1 + periodic.1) * squaredExponential.1)	-116	0.499
((periodic.1 * squaredExponential.1) + rationalQuadratic.1)	-115	0.518
((constant.1 + periodic.1) * squaredExponential.1)	-115	0.537
((rationalQuadratic.1 + squaredExponential.1) * periodic.1)	-114	0.494
(periodic.1 * (rationalQuadratic.1 * squaredExponential.1))	-114	0.499
((constant.1 + squaredExponential.1) * periodic.1)	-112	0.553
((periodic.1 * squaredExponential.1) + constant.1)	-112	0.553
(periodic.1 * (squaredExponential.1 * squaredExponential.2))	-112	0.553
(constant.1 * (periodic.1 * squaredExponential.1))	-112	0.559
(neuralNetwork.1 * (periodic.1 * squaredExponential.1))	-111	0.495
((squaredExponential.1 + squaredExponential.2) * periodic.1)	-110	0.544
((periodic.2 + squaredExponential.1) * periodic.1)	-108	0.548
((periodic.1 * squaredExponential.1) + genPoly_1.1)	-108	0.554
((periodic.1 * squaredExponential.1) + genPoly_2.1)	-108	0.553
(periodic.1 * (periodic.2 * squaredExponential.1))	-108	0.555
(genPoly_2.1 * (periodic.1 * squaredExponential.1))	-108	0.565
((periodic.1 * squaredExponential.1) + periodic.2)	-107	0.556
((genPoly_2.1 + squaredExponential.1) * periodic.1)	-107	0.554
((neuralNetwork.1 + squaredExponential.1) * periodic.1)	-106	0.552
((genPoly_1.1 + squaredExponential.1) * periodic.1)	-106	0.56
(genPoly_1.1 * (periodic.1 * squaredExponential.1))	-32.1	1.86

Table 5.1: BIC and RMSE for compound kernels fitted to the Mauna Loa atmospheric CO₂ concentration data. Kernels are arranged in the table by stage, and within each stage are ordered by BIC. The kernel selected at each stage is highlighted in bold. “genPoly_*n*” is a generalized polynomial kernel of degree *n*, while other kernel labels are hopefully self-explanatory. The appended number (e.g. “periodic.1”) is used to distinguish between distinct instances of the same kernel within a compound kernel.

```

# Load the Mauna Loa CO2 dataset
y <- datasets::co2
# Extract the times at which the time series was sampled
x <- time(y)

# Scale and center the data to have mean 0 and SD 1
y <- scale(y)
x <- scale(x)

# Initialise a model tree with a basis set of built-in kernels
mt <- gaussianProcess::create.model.tree.builtin()

# Perform a breadth-first search through the space of possible
# kernels
gp <- gaussianProcess::model.search(x, y, mt, return.all.models = T)

```

Listing 5.1: Code snippet for searching through the space of possible kernels to fit the Mauna Loa dataset.

5.1.3 Assessing model fit

There are a number of metrics by which one can assess model fit. The gold standard would be the Bayesian model evidence, $p(y | M) = \int p(y | \theta, M)p(\theta | M) d\theta$, but calculating this exactly is not computationally feasible. We discuss here a number of approximations to the model evidence which are implemented in `gaussianProcess`.

Laplace approximation to the posterior

The Laplace approximation to the posterior [Kuss and Rasmussen, 2005; Williams and Barber, 1998] approximates the posterior as a multivariate normal distribution centred on the log likelihood mode² whose inverse covariance matrix is the negative Hessian of the marginal log likelihood evaluated at the mode. With this approximation we can estimate the model evidence analytically as the integral of the normal approximation over the model parameters.

To derive the Laplace approximation for $L(\theta) = \log(p(y | M, \theta)p(\theta | M))$, we take a second order Taylor expansion around the mode $\hat{\theta}$

$$L(\theta) \approx L(\hat{\theta}) + (\theta - \hat{\theta})^T \nabla_{\theta} L(\hat{\theta}) + \frac{1}{2} (\theta - \hat{\theta})^T \nabla_{\theta}^2 L(\hat{\theta}) (\theta - \hat{\theta}) \quad (5.1)$$

²We estimate the log likelihood mode using the maximum a posteriori (MAP) estimate of the parameters, which coincides with the mode of the posterior.

which, since $\hat{\theta}$ is a local maximum, is equivalent to

$$L(\theta) \cong L(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^T \nabla_{\theta}^2 L(\hat{\theta})(\theta - \hat{\theta}) \quad (5.2)$$

and from this we see that the posterior marginal likelihood can be approximated by a normal distribution as described above

$$p(y | M, \theta)p(\theta | M) = \exp(L(\theta)) \quad (5.3)$$

$$\cong \exp\left(L(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^T \nabla_{\theta}^2 L(\hat{\theta})(\theta - \hat{\theta})\right) \quad (5.4)$$

$$= p(y | M, \hat{\theta})p(\hat{\theta} | M) \exp\left(\frac{1}{2}(\theta - \hat{\theta})^T \nabla_{\theta}^2 L(\hat{\theta})(\theta - \hat{\theta})\right) \quad (5.5)$$

Solving the Gaussian integral, we find that the Laplace approximation for the log model evidence for a Gaussian process with n hyperparameters and a kernel matrix K is:

$$\log p(y | M) = \log \left(\int p(y | M, \hat{\theta})p(\hat{\theta} | M) \exp\left(\frac{1}{2}(\theta - \hat{\theta})^T \nabla_{\theta}^2 L(\hat{\theta})(\theta - \hat{\theta})\right) \right) \quad (5.6)$$

$$= L(\hat{\theta}) + \log \left(\int \exp\left(\frac{1}{2}(\theta - \hat{\theta})^T \nabla_{\theta}^2 L(\hat{\theta})(\theta - \hat{\theta})\right) \right) \quad (5.7)$$

$$= L(\hat{\theta}) - \frac{1}{2} \log \left| -\nabla_{\theta}^2 L(\hat{\theta}) \right| + \frac{n}{2} \log 2\pi \quad (5.8)$$

$$= L(\hat{\theta}) - \frac{1}{2} \log |K^{-1}| + \frac{n}{2} \log 2\pi \quad (5.9)$$

The model which maximises this quantity is selected.

Bayesian information criterion (BIC)

The Bayesian information criterion (BIC) (also known as the Schwarz criterion) is a further approximation of the Laplace approximation [Schwarz et al., 1978]. To derive it, one rewrites the Hessian in terms of the Fisher information matrix $1/N \nabla_{\theta}^2 L$, then assumes the sample size N is significantly larger than the number of parameters n , dropping terms which do not depend on N . A derivation can be found in Neath and Cavanaugh [2012]. The BIC for a model with n parameters, N observations, and a log likelihood of $L(\hat{\theta})$ at the MAP estimate for the parameters is defined as

$$\text{BIC} = n \cdot \log(N) - 2L(\hat{\theta}) \quad (5.10)$$

The model which minimizes the BIC is selected.

5.1.4 Kernels implemented

Our R package `gaussianProcess` implements the following kernels out of the box:

- Squared exponential;
- ARD;
- Inverse ARD;³
- Rational quadratic;
- Periodic;
- Constant;⁴
- Neural network;
- Linear;
- Homogeneous polynomial;
- Generalised polynomial;
- Random partition kernels [Davies and Ghahramani, 2014].

Details of all of the kernels listed above can be found in Section 3.2.3. To maximise performance, all of the kernels above are implemented in C [Kernighan and Ritchie, 1988] and imported as R functions using `Rcpp` [Eddelbuettel and Balamuta, 2017], along with functions for computing their analytic gradients and Hessians. Using model trees, any kernel which is composed of sums and products of the built-in kernels can be represented and computed efficiently. Additional kernels can be written by the user and used in model trees, and `gaussianProcess` includes functions for creating Kernel objects from a kernel function which will create functions for numerically estimating the gradient and Hessian if no function is provided for computing the analytic results.

For all kernels, hyperparameter priors can be specified, with normal and uniform priors built in to the package.

³As per ARD, but optimised against the inverse length-scales to allow length-scales to be set to zero.

⁴ $k(x, x') = C$

5.1.5 Comparison to other R packages for fitting Gaussian processes

An overview of R packages for fitting Gaussian processes is available in Erickson et al. [2018], although it does not include `kernlab`, which is arguably the closest to providing the functionality in `gaussianProcess`. A brief overview of these packages is given below. None of the packages has functionality equivalent to the way `gaussianProcess` models compound kernels, and aside from `kernlab` no other R packages allow optimization using custom kernels.

GPfit `GPfit` [MacDonald et al., 2015] can fit GPs squared exponential and Matérn kernels, without the option to specify custom kernels. It is designed for fitting GPs to simulator output, and is therefore best suited to modelling noiseless data. It attempts to make the noise variance as small as possible whilst retaining computational stability.

kernlab `kernlab` [Karatzoglou et al., 2004] can fit a wide range of kernel functions out of the box, and the user can specify custom kernels by supplying their own kernel functions, but no functionality for fitting compound kernels.

mlegp `mlegp` [Dancik and Dorman, 2008] is designed to fit GPs with heteroscedastic noise. The documentation does not specify a kernel, but it appears to use the standard squared exponential kernel.

DiceKriging `DiceKriging` [Roustant et al., 2012], like `GPfit`, is designed for modelling simulator output. It provides a range of functionality, and includes a limited set of kernels: squared exponential, exponential, Matérn, and power-exponential. The user can provide a custom kernel function, but must supply fixed kernel hyperparameters, since `DiceKriging` will not perform hyperparameter estimation in this case.

laGP `laGP` [Gramacy, 2016] is designed for handling large datasets, and uses an approximate method for optimization and prediction.

5.1.6 Caching results in R to improve performance

Performing GP regression, particularly when assessing a set of model trees, frequently involves repeatedly calling computationally expensive functions with the same parameters, often in different functions which have no direct oversight of other

functions which have made the same call. To address this, we developed an R package `cacheMan` (available online at <https://github.com/mattdneal/cacheMan>) to perform caching of expensive function calls transparently to the calling function.

Rather than using the global environment, which is often used for caching results between functions, the cache is stored in a `Cache` object which is passed as a parameter to functions, and updates through a side effect of the `cacheMan` functions. The `Cache` object can also be persisted to disk by saving the `Cache` object using R's `save` function. Cached results are stored per-function against a hash of the function argument values. `cacheMan` also records the average time taken for cache hits and cache misses per function, and calculates the average amount of time saved (or additional time taken by unnecessary cache checks). This makes it a simple exercise to assess whether a function's performance is being helped or hindered by caching its results. An example use of `cacheMan` is shown in listing 5.2.s

At the time when we wrote `cacheMan`, no packages which provided this functionality were available on CRAN, the official repository for R packages. Since that time, a number of packages have been published which provide similar functionality, notably `memoise` [Wickham et al., 2017], and `R.cache` [Bengtsson, 2018], which both offer persistence of the cache to disk as well as caching of results in memory.

5.2 The GPLVM package

The `GPLVM` package implements a number of GPLVM variants: standard GPLVM [Lawrence, 2004], sparse GPLVM using the informative vector machine (IVM) [Herbrich et al., 2003], back-constrained GPLVM [Lawrence and Quiñonero-Candela, 2006], and discriminative GPLVM [Urtasun and Darrell, 2007]. It also implements SGPLVM and the variants described in Chapter 4: exact SGPLVM, back-constrained SGPLVM, approximate back-constrained SGPLVM, as well as SGPLVM with a discriminative prior. Listing 5.3 and listing 5.4 show example code for using the `GPLVM` package to fit GPLVM and SGPLVM models. This section covers details of the implementation of the `GPLVM` package.

5.2.1 Approximating a Hessian vector product

As described in Section 4.2.4, calculating the update step for the SMD algorithm (one of the SGD algorithms used to optimize approximate SGPLVM) requires the calculation of the product of the previous update vector \mathbf{v}_{t-1} with the Hessian of

```

> library(cacheMan)
>
> cache <- create_cache()
>
> # An inefficient prime checker
> prime = function(n) {
+ x <- 1
+ for (i in 1:n) x <- (x * i) %% n
+ x == -1
+ }
>
> # The second call with the same input fetches the result from the
> # cache
> system.time(cached_call(prime, 10^8, cache=cache))
  user system elapsed
14.315 0.016 14.353
> system.time(cached_call(prime, 10^8, cache=cache))
  user system elapsed
  0 0 0
>
> # Printing the cache returns a summary of the cache performance,
> # both overall and broken down by function
> print(cache)
prime
  cache_hits : 1
  cache_hit_time : 0
  cache_misses : 1
  cache_miss_time : 0.001
  function_call_time : 14.352
  time_saved : 14.352
  Time cost of caching: 0.001
  Time delta (saved - cost): 14.351

Totals:
  cache_hits : 1
  cache_hit_time : 0
  cache_misses : 1
  cache_miss_time : 0.001
  function_call_time : 14.352
  time_saved : 14.352
  Time saved through caching: 14.352
  Time cost of caching: 0.001

```

Listing 5.2: Example of using cacheMan to cache the results of an expensive function call.

```

# Load the iris dataset
X <- datasets::iris[,1:4]

# Fit a 2D exact GPLVM to X
exact_gplvm_2d <- GPLVM::fit.gplvm(X, q=2)

# Fit a back-constrained 2D GPLVM to X
# First, we select a lengthscale for the backconstraint using the
# heuristic described in Chapter 4
K.bc.l <- select.bc.l.centile(X)
# Then, fit the model
backconstrained_gplvm_2d <- GPLVM::fit.bcgplvm(
X,
q=2,
K.bc.l=K.bc.l
)

```

Listing 5.3: Example code for fitting various GPLVM models using the GPLVM package

```

# Create a matrix (S) which defines the structure space
# coordinates of each variable.
S.mat <- matrix(0, 5, 5)
for (i in 1:nrow(S.mat)) S.mat[i, ] <- i
S <- cbind(as.numeric(S.mat), as.numeric(t(S.mat)))
# Generate the structure and latent space kernels. Z is the
# latent space coordinates of the data.
K_S <- GPLVM::gplvm.SE(S, 3, 1, 1E-6)
K_Z <- GPLVM::gplvm.SE(Z, 0.4, 1, 1E-6)

# Sample data from the SGPLVM defined by K_S, K_Z and Z.
data <- GPLVM::sample.from.model(Z, 5, K_S, K_Z)

# Reshape the data into a 3D array
data.array <- array(data, c(nrow(Z),5,5))

# Fit an SGPLVM model to the data.
sgplvm <- GPLVM::fit.lsa_bcsplvm(data.array
iterations=10000,
points.in.approximation=500)

```

Listing 5.4: Example code for sampling data from an SGPLVM, and subsequently fitting an SGPLVM to the sampled data, using the GPLVM package, in the same manner as is seen in Chapter 4.

the log marginal likelihood at the current step, \mathbf{H}_t . In Section 4.2.4 we described a process for approximating this value without calculating the Hessian directly, resulting in eq. (4.19), which we reproduce below:

$$\mathbf{H}_t \mathbf{v}_{t-1} = \left. \frac{\partial}{\partial r} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_{t-1} + r \mathbf{v}_{t-1}) \right|_{r=0}$$

In GPLVM we use the jacobian function from the numDeriv package [Gilbert and Varadhan, 2016] to approximate this value numerically.

5.2.2 Probabilistic estimation of the trace of a matrix product

When optimizing SGPLVM without approximation it is necessary to calculate the trace of the matrix $\mathbf{K}_S^{-1} \frac{\partial}{\partial \theta_i}(\mathbf{K}_S)$ for each parameter θ_i , as per eq. (3.49). For numerical stability, it is better to solve the system $\mathbf{K}_S \mathbf{A} = \frac{\partial}{\partial \theta_i}(\mathbf{K}_S)$ for \mathbf{A} than calculate the product directly by calculating \mathbf{K}_S^{-1} . When \mathbf{K}_S is very large this process can be very memory intensive, in the worst case causing R to page memory to disk and making the computation infeasible in a reasonable timescale. To resolve this, we use a method for probabilistically estimating the trace of a matrix similar to that described by Hutchinson [1990], except that our matrix is not symmetric. Let \mathbf{B} be an $n \times n$ matrix, and $\mathbf{u} = (u_1, \dots, u_n)^T$ be a vector of n independent samples from a random variable U with mean zero and variance σ^2 . Then

$$E[\mathbf{u}^T \mathbf{B} \mathbf{u}] = E\left[\sum_{i=1}^n \sum_{j=1}^n u_i u_j b_{ij}\right] \quad (5.11)$$

$$= \sum_{i=1}^n \sum_{j=1}^n E[u_i u_j] b_{ij} \quad (5.12)$$

$$= \sum_{i=1}^n E[u_i^2] b_{ii} + \sum_{i \neq j} E[u_i] E[u_j] b_{ij} \quad (5.13)$$

$$= \sigma^2 \sum_{i=1}^n b_{ii} \quad (5.14)$$

$$= \sigma^2 \text{tr } \mathbf{B} \quad (5.15)$$

and

$$E\left[\left(\mathbf{u}^T \mathbf{B} \mathbf{u}\right)^2\right] = E\left[\left(\sum_{i,j} u_i u_j b_{ij}\right) \left(\sum_{r,s} u_r u_s b_{rs}\right)\right] \quad (5.16)$$

$$= E \left[\sum_{i,j} \sum_{r,s} u_i u_j u_r u_s b_{ij} b_{rs} \right] \quad (5.17)$$

$$= \sum_{i \neq j} (b_{ij}^2 + b_{ij} b_{ji}) E[u_i^2] E[u_j^2] + \sum_{i \neq j} b_{ii} b_{jj} E[u_i^2] E[u_j^2] + \sum_i b_{ii}^2 E[u_i^4] \quad (5.18)$$

$$= \sigma^4 \sum_{i \neq j} (b_{ij}^2 + b_{ij} b_{ji}) + \sigma^4 \sum_{i \neq j} b_{ii} b_{jj} + E[U^4] \sum_i b_{ii}^2 \quad (5.19)$$

which gives a variance of

$$\text{Var}(\mathbf{u}^T \mathbf{B} \mathbf{u}) = E \left[(\mathbf{u}^T \mathbf{B} \mathbf{u})^2 \right] - E \left[(\mathbf{u}^T \mathbf{B} \mathbf{u}) \right]^2 \quad (5.20)$$

$$= \sigma^4 \sum_{i \neq j} (b_{ij}^2 + b_{ij} b_{ji}) + \sigma^4 \sum_{i \neq j} b_{ii} b_{jj} + E[U^4] \sum_i b_{ii}^2 - \sigma^4 \sum_{i,j} b_{ii} b_{jj} \quad (5.21)$$

$$= \sigma^4 \sum_{i \neq j} (b_{ij}^2 + b_{ij} b_{ji}) + \left(E[U^4] - \sigma^4 \right) \sum_i b_{ii}^2 \quad (5.22)$$

which differs slightly from the result given in Hutchinson [1990] since they restrict \mathbf{B} to be symmetric.

As in Hutchinson [1990], we draw u_i from $\{-1, 1\}$ with

$$p(u_i = -1) = p(u_i = 1) = \frac{1}{2} \quad (5.23)$$

In this case $\sigma^2 = 1$ and therefore

$$E[\mathbf{u}^T \mathbf{B} \mathbf{u}] = \text{tr } \mathbf{B} \quad (5.24)$$

$$\text{Var}(\mathbf{u}^T \mathbf{B} \mathbf{u}) = \sum_{i \neq j} (b_{ij}^2 + b_{ij} b_{ji}) \quad (5.25)$$

Rather than calculating the variance analytically, in our implementation we estimate the variance from the sample, adding samples in batches of 128 until the ratio of the standard error to the mean is below a given threshold (0.1 by default).

5.2.3 Piecewise exact calculation of the trace of a matrix product

As an alternative to the process outlined in Section 5.2.2, the trace of $\mathbf{K}_S^{-1} \frac{\partial}{\partial \theta_i} (\mathbf{K}_S)$ can be calculated piecewise by subdividing $\frac{\partial}{\partial \theta_i} (\mathbf{K}_S)$ into disjoint submatrices \mathbf{B}_i ,

where \mathbf{B}_i is composed of columns $m(i-1)+1$ to mi for i from 1 to $\lceil n/m \rceil$.⁵ We then solve $\mathbf{K}_S \mathbf{A} = \mathbf{B}_i$ for \mathbf{A} , and sum the elements on the diagonal of \mathbf{A} which begins at row $m(i-1)+1$. This process allows us to limit the maximum memory required when computing the trace, ensuring the calculation fits in memory throughout.

5.2.4 Priors on latent space coördinates

Because GPLVMs and SGPLVMs preserve differences between points in the observed space in the latent space, it is sometimes the case that points which are dissimilar are initialised close together in the latent space, and for some optimization algorithms this can lead to points being “ejected” from the main mass of points in latent space, due to a very large gradient pushing the two points apart. Once a point is several multiples of the latent space length-scale from any other points, it will tend not to move from that position, since its interactions with other points are very small, and therefore the impact of small changes in its location are also small.

To combat this, it is helpful to apply a weak normal prior to the latent coördinates. This has minimal impact on the learned structure when points are within a few length-scales and therefore close enough to interact, but will gradually move ejected points back to a position where they have non-negligible interactions with other points. In our implementation we default to a normal prior on the latent coördinates with a variance of 100. There is also the option to apply an improper uniform prior

5.2.5 Representing structured data in R

We use two distinct schema for representing the structured data modelled by SGPLVM in the GPLVM package. For inputting the observed data into the various functions for fitting SGPLVMs we use multi-dimensional arrays, as created using the `array` function in base R. The first dimension of the array indexes the samples, and the remaining dimensions of the array index the various dimensions of the structure space. By way of example, FAIMS data is stored in a five-dimensional array, with the dimensions indexing the sample ID, compensation voltage, dispersion field strength, polarity, and run number. Missing data is input by setting the missing entries to NA.

For the approximate algorithm, we need to be able to select a subset of entries of the input observed data array. To facilitate this, we flatten the multi-dimensional array into a matrix where each row contains a vector giving the coördinates, and

⁵With the final submatrix composed of fewer columns if necessary.

a vector of the corresponding entries in the array. This makes it trivial to select subsets of the entries in the array, and to calculate the latent and structure kernels for the active set used for approximation.

5.2.6 Efficiently sampling from an SGPLVM prior

Given a latent space, a kernel over that latent space, and a structure kernel, it is relatively straightforward to draw samples from an SGPLVM prior defined over that space. Given a positive definite $n \times n$ matrix \mathbf{K}_Z and a positive definite $k \times k$ matrix \mathbf{K}_S , create the $n \times k$ matrix $\widetilde{\mathbf{X}}$ where $x_{ij} \sim \mathcal{N}(0, 1)$.

Since \mathbf{K}_Z and \mathbf{K}_S are positive definite, they admit Cholesky decompositions $\mathbf{K}_Z = \mathbf{R}_Z^T \mathbf{R}_Z$ and $\mathbf{K}_S = \mathbf{R}_S^T \mathbf{R}_S$, where \mathbf{R}_Z and \mathbf{R}_S are upper triangular matrices. Then

$$\mathbf{X} = \mathbf{R}_Z^T \widetilde{\mathbf{X}} \mathbf{R}_S \quad (5.26)$$

is a sample from the SGPLVM defined by \mathbf{K}_Z and \mathbf{K}_S . This is derived from the fact that the noise-free SGPLVM is equivalent to the matrix normal distribution, $\widetilde{\mathbf{X}}$ is a sample from $\mathcal{MN}(0, \mathcal{I}, \mathcal{I})$, and by the properties of linear transformations of matrix normal distributions,

$$\mathbf{R}_Z^T \widetilde{\mathbf{X}} \mathbf{R}_S \sim \mathcal{MN}(0, \mathbf{R}_Z^T \mathbf{R}_Z, \mathbf{R}_S^T \mathbf{R}_S) = \mathcal{MN}(0, \mathbf{K}_Z, \mathbf{K}_S) \quad (5.27)$$

as required.

5.2.7 Discriminative priors for SGPLVM

As formulated in Chapter 4, GPLVM and SGPLVM are generative models, producing a latent representation which captures the latent structure in the data. If GPLVM is being used as the first step in a classification task, it would be useful to target the latent space to capture latent structure which is relevant to the classification task. One method for incorporating class knowledge into the GPLVM prior is Discriminative GPLVM [Urtasun and Darrell, 2007].⁶ Instead of placing an uninformative uniform or a normal prior on \mathbf{Z} , one enforces a prior derived from Fisher’s linear discriminant [Fisher, 1936], which places a higher probability on values of \mathbf{Z} which maximize the distance between classes whilst minimizing the distance within classes. We extended this approach to SGPLVM.

⁶A typo exists in Urtasun and Darrell [2007] whereby the definitions for \mathbf{S}_w and \mathbf{S}_b are switched in equations (10) and (11). Here we use the correct definitions as given in Sugiyama [2006], however the general approach of Urtasun and Darrell [2007] is valid once the typo is accounted for.

Given a latent representation \mathbf{Z} , with each point belonging to one of L classes, we apply a prior

$$p(\mathbf{Z}) = K \exp\left(-\frac{1}{\sigma_d^2} J^{-1}\right) \quad (5.28)$$

where σ^2 is a global scaling of the prior, K is a normalizing constant, and

$$J(\mathbf{Z}) = \text{tr}(\mathbf{S}_w^{-1} \mathbf{S}_b) \quad (5.29)$$

weights for latent representations which maximise the between-class separability while minimizing the within-class variability. \mathbf{S}_w and \mathbf{S}_b are respectively the within- and between-class matrices:

$$\mathbf{S}_w = \frac{1}{N} \sum_{i=1}^L \sum_{k=1}^{N_i} \left(\mathbf{z}_k^{(i)} - \mathbf{M}_i\right) \left(\mathbf{z}_k^{(i)} - \mathbf{M}_i\right)^T \quad (5.30)$$

$$\mathbf{S}_b = \sum_{i=1}^L \frac{N_i}{N} (\mathbf{M}_i - \mathbf{M}_0) (\mathbf{M}_i - \mathbf{M}_0)^T \quad (5.31)$$

where $\mathbf{Z}^{(i)} = (\mathbf{z}_1^{(i)}, \dots, \mathbf{z}_{N_i}^{(i)})^T$ is the matrix composed of the N_i training points in class i , \mathbf{M}_i is the mean of the elements of class i , and \mathbf{M}_0 is the mean of all points in the training set.

Figure 5.3 shows the effect of the discriminative prior on the learned latent representation of Dataset 1 of the close-clustered datasets described in Chapter 4. A range of prior strengths are shown, along with a latent representation learned under a uniform prior. Selecting an appropriate prior strength is a heuristic process, using the rule of thumb that the prior strength should be increased until some change in the latent representation is observed, but without the effects of the prior dominating the latent representation. Increasing prior strength leads as expected to increasing separation of the classes in the latent representation. The figure also shows the effect when the discriminative prior dominates the likelihood, with entirely distinct clusters which are not representative of the true latent space.

5.3 Summary

We have presented two R packages which provide novel capabilities not currently available in other publicly available R packages. The `gaussianProcess` package offers flexible, fast learning of Gaussian process regression models, with a wide range of kernels and an elegant system for defining, fitting, and generating arbitrary

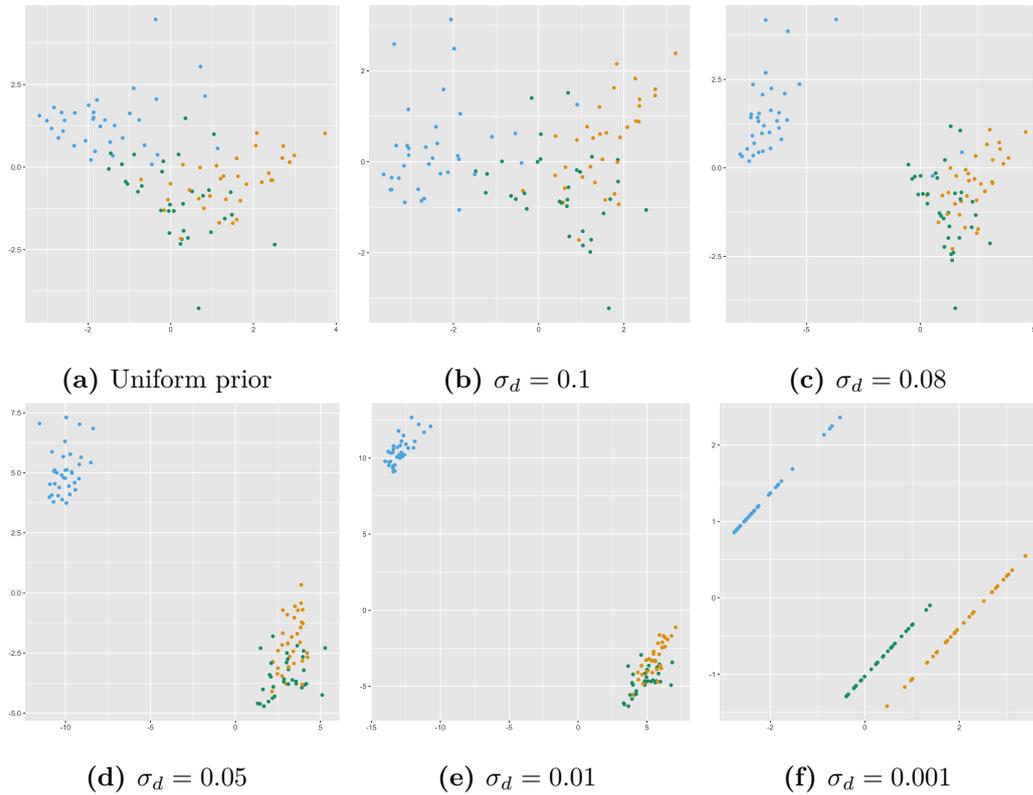


Figure 5.3: The effect of discriminative priors on the learned latent representation in SGPLVM. Shown are six latent representations of Dataset 1 of the close-clustered artificial data used in Chapter 4 obtained by either SGPLVM with a uniform prior on the latent space **(a)**, or a discriminative prior with different values of σ_d **(b–f)**. Decreasing σ_d increases the strength of the prior, leading to increased emphasis on separation of the classes. **(f)** shows the effect of the discriminative prior dominating the likelihood, leading to pathological behaviour when optimizing the latent representation.

compound kernels. The **GPLVM** package provides functions for fitting both a range of GPLVM variants, and also implements the novel SGPLVM framework detailed in Chapter 4.

Chapter 6

Conclusion

In this thesis we have considered urinary VOCs, measured using FAIMS, as a screening test for the early detection of cancer. We used machine learning techniques to build classifiers based on FAIMS data for the detection of colorectal cancer and a number of related conditions, and developed publicly available software for performing this analysis. We also presented a novel Bayesian non-linear dimensionality reduction technique, SGPLVM, along with a stochastic optimization algorithm which makes the technique computationally feasible on large datasets, and associated software implementing Gaussian process regression, GPLVM, and SGPLVM.

In Chapter 2, we considered the use of urinary VOCs as a screening test for the early detection of cancer. Urinary VOCs show promise at detecting colorectal cancer, but are significantly outperformed by the current screening method, FIT. However, adjusting for compliance rate differences between faecal and urine testing offers an interesting and oft-overlooked alternative perspective where the difference in performance is significantly less stark. More generally, urinary VOCs show promise as a novel biomarker for disease, and may well have more utility when applied to detecting cancers for which there is no current screening programme. Unfortunately, testing this hypothesis is both more difficult and more expensive than testing it within an existing screening framework—recruiting a representative cohort of patients is much easier when they are already presenting for a screening programme, and gathering outcome data is similarly more straightforward when the pathway for diagnosis is already established. On a more encouraging note, urinary VOCs showed an excellent ability to perform pre-symptomatic detection of complications resulting from colorectal resection for the treatment of cancer, with an AUC of 0.91 (95% CI 0.81, 1.0).

Chapter 2 also introduced our R package `FAIMSToolkit` for analysing FAIMS data. We believe it offers a number of benefits for researchers in this field. It makes the use of modern machine learning techniques accessible to a wider audience, and through the integration of the `caret` package will do hyperparameter tuning transparently to the end user in a way which avoids introducing bias into the reported results. For users with more grounding in machine learning who may have particular techniques in mind, `FAIMSToolkit` is easily extensible to incorporate new learning algorithms without changes to the underlying source code.

Our package removes a number of possible sources of error which we have seen when working with FAIMS data, including in some published studies. The analysis pipeline `FAIMSToolkit` implements prevents accidentally leaking class information from the test set to the training algorithm by ensuring that cross-validation is applied appropriately to every step of the analysis, removing a possible positive bias in results. The automation of assigning autosampled data to labelled samples, and the detection and removal of “blank” between-sample runs is both a tremendous time saving and also removes the risk of mislabelling of data, which we have encountered when autosampled data has been manually labelled. And the inclusion of outlier detection functions replaces the previously prevalent method of visually inspecting the output data for incongruous or erroneous plumes.

SGPLVM, the model introduced in Chapter 4, represents a novel Bayesian model for data with structured correlations between variables, of which FAIMS data is an example. Unfortunately, exact optimization of the model parameters requires that one repeatedly invert the Kronecker product of two matrices, which for even modestly-sized datasets is computationally infeasible due to the size of the resulting matrix. To resolve this, we developed a stochastic optimization method based on SGD, which allows approximate optimization of the model parameters in reasonable time on a standard modern workstation.

The results in Chapter 4 provide encouraging validation that SGPLVM provides improved latent space recovery compared to GPLVM for data which fits the underlying model. It also explores the ability of SGPLVM to handle partially observed data points which, although a direct result of the use of the structured prior to collapse the multiple Gaussian processes in a GPLVM into the single Gaussian process of an SGPLVM, was not a property of the model which was in mind when we were developing it. SGPLVM proves to be highly resilient to partially observed data, only

losing fidelity in the recovered latent space when roughly 70% of data points were missing.

Despite the success when modelling artificial data, SGPLVM did not recover sufficient detail when applied to FAIMS data to lead to improved classification performance compared to our existing pipeline from Chapter 2. However it did perform better than alternative methods of dimensionality reduction at recovering a useful low-dimensional latent representation, and so may still have a place as a visualisation method for FAIMS datasets.

Chapter 5 provides implementation details for our two R packages for GPs and SGPLVM. The former, `gaussianProcess`, implements Gaussian process regression in R, and offers a number of features which are not available in other publicly available R packages. Specifically, it efficiently implements a range of kernels in C, and also includes a framework for generating arbitrary compound kernels and optimizing the resulting GPs. In addition to this, we developed a process for performing automatic model selection using a width-first search based on Bayesian model selection criteria. The latter package, `GPLVM`, provides an R implementation of GPLVM and a number of extensions, as well as implementing SGPLVM with both exact and stochastic optimization. While software for fitting GPLVMs is available for a number of languages,¹ to our knowledge our package is the only publicly available package which provides an implementation of GPLVM with exact optimization,² as well as the only implementations of back-constrained and discriminative GPLVM in R.

Two obvious further developments for SGPLVM present themselves at this time. The first is to implement an approximate version of SGPLVM which is optimized using variational Bayesian inference, as has been done with Gaussian processes [Tran et al., 2015] and GPLVM [Titsias and Lawrence, 2010]. If the results of this mirror those for GPs and GPLVMs, this can be expected to improve both the computation time and the learned latent space. The second avenue to explore is using the latent representation as the input space for a Gaussian process classifier (GPC), and optimizing both the latent representation and the GPC concurrently, with the intention that the GPC would inform the positioning of the latent points so as to maximise the utility of the latent representation for classification. Using

¹For example, GPLVM packages are available for Python, C++, and MATLAB.

²An implementation of Bayesian GPLVM is available at <https://github.com/SheffieldML/vargplvm>.

back-constraints would then allow for new samples to be quickly placed into the latent space, and then classified using the trained GPC.

Looking forward, VOCs still show promise as a pre-symptomatic screening test for early cancer detection. FAIMS analysis is a reproducible, low-cost, non-invasive method of testing, and can work with sample mediums with good patient acceptability. Unfortunately, its utility as a screening test for colorectal cancer is limited because of the effectiveness of the existing screening test, FIT. More work is required to determine the effectiveness of VOCs for detecting other cancer types where no adequate screening test is available.

Appendix A

TWW Study JAGS Models

A.1 Logistic regression

```
# NPC Continuous clinical predictors
# NS Num samples
# predictorCont matrix of continuous clinical predictors
# disease vector of disease status, by disease index

model {
  # Likelihood:
  for (sampIdx in 1:NS) {
    disease[sampIdx] ~ dbern( pDisease[sampIdx] )
    #Scaling and centering from Gelman pp.415–6
    pDiseaseLatent[sampIdx] ~ dnorm(alpha
      + sum(betaCont
        * (predictorCont[sampIdx, ] -
          predictorCont.mean)
        / (2 * predictorCont.sd)
      ),
      1)
    logit(pDisease[sampIdx]) <- pDiseaseLatent[sampIdx]
  }
  for (predCIIdx in 1:NPC) {
    predictorCont.mean[predCIIdx] <- mean(predictorCont[, predCIIdx])
    predictorCont.sd[predCIIdx] <- sd(predictorCont[, predCIIdx])
    for (sampIdx in 1:NS) {
      predictorCont[sampIdx, predCIIdx] ~ dnorm(mu.pc[predCIIdx],
        tau.pc[predCIIdx])
    }
  }
}
```

```

    }
    mu.pc[predCIIdx] ~ dnorm(0, 10E-3 * tau.pc[predCIIdx])
    tau.pc[predCIIdx] ~ dgamma(10E-3, 10E-3)
  }
  # Prior:
  #alpha prior from Gelman p.416
  alpha ~ dt(0, 1/sqrt(10), 1)
  #beta prior from Gelman p.416
  for (predCIIdx in 1:NPC) {
    betaCont[predCIIdx] ~ dt(0, 1/sqrt(2.5), 1)
  }
}

```

A.2 Logistic regression with interactions

```

# NPC Continuous clinical predictors
# NPB Binary clinical predictors
# NS Num samples
# predictor matrix of predictors
# binaryPredictor 1 if predictor is binary, 0 otherwise.
# disease vector of disease status, by disease index
data {
  NP <- NPC + NPB
  numCoef <- NP * (NP + 1) / 2
}
model {
  # Likelihood:
  for (sampIdx in 1:NS) {
    disease[sampIdx] ~ dbern( pDisease[sampIdx] )
    #Scaling and centering from Gelman pp.415-6
    predictorTrans[sampIdx, 1:NPC] <- (predictorCont[sampIdx, ] -
      predictorCont.mean)
      / (2 * predictorCont.sd)

    for (i in 1:NP) {
      predictorVector[sampIdx, i] <- predictorTrans[sampIdx, i]
    }

    #Interaction terms
    for (i in 1:(NP - 1)) {
      for (j in (i+1):NP) {
        predictorVector[sampIdx, NP + (NP*(NP-1)/2) -
          (NP-i+1)*((NP-i+1)-1)/2 + j - i] <-

```

```

        predictorTrans[sampIdx, i] * predictorTrans[sampIdx, j]
    }
}

pDiseaseLatent[sampIdx] ~ dnorm(alpha
                                + sum(beta *
                                        predictorVector[sampIdx, ]),
                                1)

logit(pDisease[sampIdx]) <- pDiseaseLatent[sampIdx]
}
for (predCIIdx in 1:NPC) {
  predictorCont.mean[predCIIdx] <- mean(predictorCont[, predCIIdx])
  predictorCont.sd[predCIIdx] <- sd(predictorCont[, predCIIdx])
  for (sampIdx in 1:NS) {
    predictorCont[sampIdx, predCIIdx] ~
      dnorm(mu.pc[predCIIdx], tau.pc[predCIIdx])
  }
  mu.pc[predCIIdx] ~ dnorm(0, 10E-3 * tau.pc[predCIIdx])
  tau.pc[predCIIdx] ~ dgamma(10E-3, 10E-3)
}
# Prior:
#alpha prior from Gelman p.416
alpha ~ dt(0, 1/sqrt(10), 1)
#beta prior from Gelman p.416
for (coefIdx in 1:numCoef) {
  beta[coefIdx] ~ dt(0, 1/sqrt(2.5), 1)
}
}

```

A.3 Robust logistic regression

```

# NPC Continuous clinical predictors
# NS Num samples
# predictorCont matrix of continuous clinical predictors
# disease vector of disease status, by disease index

model {
  # Likelihood:
  for (sampIdx in 1:NS) {
    disease[sampIdx] ~ dbern( pDisease[sampIdx] )
    #Scaling and centering from Gelman pp.415-6
  }
}

```

```

pDiseaseLatent[sampIdx] ~ dt(alpha
                             + sum(betaCont
                                   * (predictorCont[sampIdx, ] -
                                       predictorCont.mean)
                                   / (2 * predictorCont.sd)
                                   ),
                             1,
                             4)
logit(pDisease[sampIdx]) <- pDiseaseLatent[sampIdx]
}
for (predCIdx in 1:NPC) {
  predictorCont.mean[predCIdx] <- mean(predictorCont[, predCIdx])
  predictorCont.sd[predCIdx] <- sd(predictorCont[, predCIdx])
  for (sampIdx in 1:NS) {
    predictorCont[sampIdx, predCIdx] ~ dnorm(mu.pc[predCIdx],
                                             tau.pc[predCIdx])
  }
  mu.pc[predCIdx] ~ dnorm(0, 10E-3 * tau.pc[predCIdx])
  tau.pc[predCIdx] ~ dgamma(10E-3, 10E-3)
}
# Prior:
#alpha prior from Gelman p.416
alpha ~ dt(0, 1/sqrt(10), 1)
#beta prior from Gelman p.416
for (predCIdx in 1:NPC) {
  betaCont[predCIdx] ~ dt(0, 1/sqrt(2.5), 1)
}
}

```

A.4 Independent robust mixture model

```

# NB Biomarkers
# ND Diseases
# NS Num samples
# biomarker matrix of biomarker data
# disease vector of disease status, by disease index
# overdispersion factor
data {
  # Jeffrey's prior for categorical dist.
  for (i in 1:ND) {
    alpha[i] <- 1/2
  }
}

```

```

}
model {
  # Likelihood:
  for (sampIdx in 1:NS) {
    for (bioIdx in 1:NB) {
      biomarker[sampIdx, bioIdx] ~ dt(mu[bioIdx, sampIdx],
                                     tau[bioIdx, sampIdx],
                                     biomarkerNu)

      mu[bioIdx, sampIdx] <- muOfClust[bioIdx, disease[sampIdx]]
      tau[bioIdx, sampIdx] <- tauOfClust[bioIdx, disease[sampIdx]]
    }
    disease[sampIdx] ~ dcat( pDisease )
  }
  # Prior:
  pDisease ~ ddirch(alpha)
  for (disIdx in 1:ND) {
    for (bioIdx in 1:NB) {
      muOfClust[bioIdx, disIdx] ~ dnorm( 0, 1.0E-3)
      tauOfClust[bioIdx, disIdx] ~ dgamma(1.0E-3, 1.0E-3)
    }
  }
}

```

A.5 Independent robust mixture model with implicit subclasses and censoring

```

# NB Biomarkers
# ND Diseases
# NS Num samples
# biomarker matrix of biomarker data
# disease vector of disease status, by disease index
# biomarkerNu dispersion parameter for the biomarkers
# numSubClasses number of sub classes
# biomarkerNu Dispersion for biomarker
# biomarkerCensorClass -1 for left-censored, 0 for uncensored,
# 1 for right-censored
data {
  for (i in 1:NS) {
    zeroes[i] <- 0
  }

  for (i in 1:ND) {

```

```

    alphaD[i] <- 1/2
  }
  for (i in 1:numSubClasses) {
    alphaS[i] <- 1/2
  }
}
model {
  # Likelihood:
  for (sampIdx in 1:NS) {
    for (subCIdx in 1:numSubClasses) {
      for (bioIdx in 1:NB) {
        # switch likelihood contribution depending on whether
        # data is censored or not.
        phiBioSub[subCIdx, bioIdx, sampIdx] <-
          ifelse(biomarkerCensorClass[sampIdx, bioIdx] == 0,
            logdensity.t(biomarker[sampIdx, bioIdx],
              mu[sampIdx, bioIdx, subCIdx],
              tau[sampIdx, bioIdx, subCIdx],
              biomarkerNu),
            ifelse(biomarkerCensorClass[sampIdx, bioIdx] == -1,
              log(pt(biomarker[sampIdx, bioIdx],
                mu[sampIdx, bioIdx, subCIdx],
                tau[sampIdx, bioIdx, subCIdx],
                biomarkerNu)),
                log(1 - pt(biomarker[sampIdx, bioIdx],
                  mu[sampIdx, bioIdx, subCIdx],
                  tau[sampIdx, bioIdx, subCIdx],
                  biomarkerNu))))))
        mu[sampIdx, bioIdx, subCIdx] <-
          muOfClust[bioIdx,
            subCIdx,
            disease[sampIdx]]
        tau[sampIdx, bioIdx, subCIdx] <-
          tauOfClust[bioIdx,
            subCIdx,
            disease[sampIdx]]
      }
      phiSub[subCIdx, sampIdx] <-
        sum(phiBioSub[subCIdx, , sampIdx]) +
        log(pSampSubClass[subCIdx, sampIdx])
    }
  }
}

```

```

# Numerically stable LogSumExp calculation:
maxPhi[sampIdx] <- max(phiSub[, sampIdx])

for (subCIIdx in 1:numSubClasses) {
  phiDiffExp[subCIIdx, sampIdx] <-
    exp(phiSub[subCIIdx, sampIdx] -
      maxPhi[sampIdx])
}
phi[sampIdx] <-
  -(maxPhi[sampIdx] + log(sum(phiDiffExp[, sampIdx])))
zeroes[sampIdx] ~ dpois(phi[sampIdx])

pSampSubClass[1:numSubClasses, sampIdx] <-
  pSubClass[1:numSubClasses,
    disease[sampIdx]]
disease[sampIdx] ~ dcat(pDisease)

}

# Prior:
pDisease ~ ddirch(alphaD)
for (i in 1:ND) {
  pSubClass[1:numSubClasses, i] ~ ddirch(alphaS)
}
for (disIdx in 1:ND) {
  for (bioIdx in 1:NB) {
    for (subCIIdx in 1:numSubClasses) {
      muOfClust[bioIdx, subCIIdx, disIdx] ~
        dnorm( 0 , 1.0E-3 )
      tauOfClust[bioIdx, subCIIdx, disIdx] ~
        dgamma( 1.0E-3 , 1.0E-3 )
    }
  }
}
}
}

```


Appendix B

FAIMS Autosampler Data Auto-splitter

B.1 readme.txt

Before you run it for the first time:

- 1) Install R by running R-3.3.2-win.exe. Make sure you install it to the default location.

Steps to run the program once R is installed:

- 1) Put the binary FAIMS data files for one run in a single folder.
- 2) Convert them using my faims_converter.bat tool.
- 3) Copy the example sampleNames.txt file into the folder, open it, delete the contents, and enter the names of the samples you have run (including blanks) into the file, with one sample per line.
- 4) Run the splitter by double clicking FAIMSSplitter.bat.
- 5) Enter the full path to the folder containing the FAIMS data when prompted.
- 6) Enter "asc" (without quotes) when prompted for an extension.
- 7) Let the program run
- 8) Enjoy your split files!

Troubleshooting:

The program will spit out words of wisdom if it hits a problem. It will also produce two graphs for debugging, showing the flow rate per run and the assigned sample id per run (example output is included in this zip file).

If a sample failed to run, causing a very long stretch of low flow rate where there should be a sample, delete the failed sample from sampleNames.txt and re-run the program.

Appendix C

FAIMSToolkit documentation

Type Package

Title Tools to aid in the analysis of FAIMS data

Version 0.2.15

Maintainer Matthew Neal <mattdneal@gmail.com>

Description Tools for ingesting and analysing FAIMS data

License MIT

LazyData TRUE

Encoding UTF-8

RoxygenNote 6.1.1

Imports caret,
pROC,
wavethresh,
tcltk,
lattice,
grid,
boot

Suggests GPLVM,
viridis

boot_auc_folds

Compute bootstraps of the mean ROC

Description

Compute the ROC curve separately for each fold, and compute the mean ROC curve, instead of pooling predictions.

Usage

```
boot_auc_folds(predictions, folds, targets, num_samples = 1000,  
alt_predictions = NULL, fixed_axis = c("sensitivity",  
"specificity"), points = seq(0, 1, by = 0.01), stratified = T)
```

Arguments

<code>predictions</code>	sample predictions
<code>folds</code>	vector of fold classification for each sample
<code>targets</code>	target values
<code>num_samples</code>	number of bootstrap samples to take
<code>alt_predictions</code>	alternative predictions to compare
<code>fixed_axis</code>	whether to fix sensitivity or specificity in the ROC curves
<code>points</code>	number of points to calculate for ROC curves

Value

bootstrapped ROC curves

<code>calc_mean_roc</code>	<i>Calculate the mean ROC curve from a list of predictors and targets</i>
----------------------------	---

Description

Calculate the mean ROC curve from a list of predictors and targets

Usage

```
calc_mean_roc(predictions_list, targets_list, points,  
fixed_axis = c("specificity", "sensitivity"), boot_index = NULL,  
return_folds = FALSE)
```

Arguments

<code>predictions_list</code>	list of predictors
<code>targets_list</code>	list of targets
<code>points</code>	points to calculate the ROC curves at
<code>fixed_axis</code>	whether <code>points</code> fixes sensitivity or specificity
<code>boot_index</code>	list of bootstrap indices for each fold. Generated automatically if NULL
<code>return_folds</code>	if TRUE, return the ROC curve and AUC for each fold as well as the mean

Value

list containing mean ROC curve and AUC

<code>checkDrift</code>	<i>Check for FAIMS drift</i>
-------------------------	------------------------------

Description

Check for FAIMS drift

Usage

```
checkDrift(faims, index = NULL, ...)
```

Arguments

<code>faims</code>	a faims object
<code>index</code>	the samples to include. NULL (the default) includes all samples
<code>...</code>	other parameters to pass to plot

Value

a list with the correlation, a 95

<code>checkFlowRate</code>	<i>Check the minimum flow rates for all the runs in a dataset</i>
----------------------------	---

Description

Check the minimum flow rates for all the runs in a dataset

Usage

```
checkFlowRate(dir, threshold = 1.9,  
filePattern = ".*[.](txt|asc)", moveTo = NULL)
```

Arguments

<code>dir</code>	Directory pat to check
<code>threshold</code>	Threshold for flow rate
<code>filePattern</code>	Regex pattern of exported FAIMS files
<code>moveTo</code>	Folder to move low flow rate files to

Value

data frame listing minimum flow rate per folder

<code>ClassifierModels</code>	<i>Run a set of classification models on input training, test data sets</i>
-------------------------------	---

Description

Run a set of classification models on input training, test data sets

Usage

```
ClassifierModels(data.train, targetValues, data.test, models,  
kFolds = 2, repeats = 5, tuneLength = 5, verbose = F)
```

Arguments

<code>data.train</code>	a data frame of training data
<code>targetValues</code>	a logical vector
<code>data.test</code>	a data frame of test data (columns must match <code>data.train</code>)
<code>models</code>	a list of <code>caret::train</code> models to run
<code>kFolds</code>	number of folds for model selection within each fold
<code>repeats</code>	number of repeats for model selection within each fold
<code>tuneLength</code>	number of parameters to tune
<code>verbose</code>	verbose output if TRUE

Value

a data frame containing prediction probabilities for each classification algorithm. These are the predicted probability of `targetValues==TRUE`

`compare_auc_bootstrap`

Compute a bootstrap of the difference between two ROC curve AUCs

Description

Compute a bootstrap of the difference between two ROC curve AUCs

Usage

```
compare_auc_bootstrap(roc_1, roc_2, rep = 10000)
```

Arguments

<code>roc_1</code>	first ROC curve (from pROC)
<code>roc_2</code>	second ROC curve (from pROC)
<code>rep</code>	number of bootstrap samples

Value

bootstrap of the difference in AUC

`convertToArray` *Convert FAIMS data to an array*

Description

Converts a FAIMS object to a FAIMSArray object, with the data stored in a 5-dimensional array. The dimensions correspond to sample ID, CV, dispersion field strength, polarity, and runNumber.

Usage

```
convertToArray(FAIMSObject, keepMatrixData = TRUE)
```

Arguments

`FAIMSObject` A FAIMS object

`keepMatrixData`

Whether to keep the matrix-shaped FAIMS data

Details

If `keepMatrixData=TRUE`, the output inherits both FAIMS and FAIMSArray. Otherwise, it only inherits FAIMSArray and will not work with functions which require FAIMS objects.

Value

A FAIMSArray object (also inheriting FAIMS if `keepMatrixData=TRUE`)

`CrossValidation` *Cross-validation for classification models*

Description

Cross-validation for classification models

Usage

```
CrossValidation(data.train, targetValues, models = c("rf", "glmnet",
"svmRadial", "svmLinear", "gbm", "nnet", "glm"), nFolds = 10,
stratified = TRUE, threshold = NULL, nKeep = NULL, SGoF = NULL,
verbose = FALSE, heatmap = FALSE, PCA = FALSE, extraData = NULL,
tuneKFolds = 2, tuneRepeats = 5, tuneLength = 5, folds = NULL,
precomputedScores = NULL)
```

Arguments

<code>data.train</code>	Training data to be divided into folds (must be a data frame)
<code>targetValues</code>	Target responses
<code>models</code>	list of <code>caret::train</code> models to train
<code>nFolds</code>	number of folds
<code>stratified</code>	TRUE for stratified folds
<code>threshold</code>	list of threshold p-values for selecting features to keep
<code>nKeep</code>	list of number of features to keep
<code>SGoF</code>	alpha for Sequential Goodness of Fit selection of features
<code>verbose</code>	TRUE for verbose output
<code>heatmap</code>	TRUE to plot a heatmap of selected features for each fold
<code>PCA</code>	TRUE to apply PCA to selected features in each fold
<code>extraData</code>	any additional data to add to training data after feature selection
<code>tuneKFolds</code>	number of folds for tuning models within each fold
<code>tuneRepeats</code>	number of repeats for tuning models within each fold
<code>tuneLength</code>	number of parameters to test when tuning models
<code>folds</code>	optionally pre-specify which samples go in which fold. Should be NULL to select folds randomly, or a vector of length <code>nrow(data.train)</code> containing values in <code>seq(nFolds)</code>
<code>precomputedScores</code>	precomputed scores of features for each fold. A sample of 100 will be tested for each fold and an error thrown in the event of discrepancies.

Value

A list of predictions for the given model combinations, ready to be passed to `CrossValRocCurves`

`CrossValRocCurves` *Compute ROC objects for the results of a cross-validation*

Description

Compute ROC objects for the results of a cross-validation

Usage

```
CrossValRocCurves(crossVal.obj)
```

Arguments

`crossVal.obj` the output from `CrossValidation`

Value

A list of `pROC::roc` objects and a summary of the results

`deleteFAIMSSample` *Delete samples from a FAIMS object*

Description

Delete samples from a FAIMS object

Usage

```
deleteFAIMSSample(FAIMSObject, deleteIndices)
```

Arguments

`FAIMSObject` a FAIMS object
`deleteIndices`
 indices to delete

Value

a FAIMS object

`denoiseFaimsData` *Remove background noise from FAIMS data*

Description

Denotes FAIMS data. First identifies pixels likely to be solely noise (mean value across all samples close to zero). Then identifies a value to subtract from all pixels by taking the `fractionToRemove` quantile of all the values in the noise pixels, and subtracts this from the absolute value of each pixels, zeroing any values which are negative after subtracting background noise.

Usage

```
denoiseFaimsData(faimsObject, fractionNoise = 0.05,  
fractionToRemove = 0.99)
```

Arguments

`fractionNoise`

The percentage of pixels to identify as "noise"

`fractionToRemove`

The quantile value of the noise pixels to subtract from all pixels

`FAIMSObject` FAIMS object

Value

a denoised FAIMS data matrix

`denoiseFaimsData.localCorr`

Remove background noise from FAIMS data using local correlations

Description

The signal in FAIMS data exhibits a high degree of local correlation. This function zeroes out a fraction of pixels with the lowest degree of local correlation.

Usage

```
denoiseFaimsData.localCorr(FAIMSObject, neighbourhoodSizeCol = 1,  
neighbourhoodSizeRow = 1, alpha = 0.05, plot = TRUE)
```

Arguments

<code>FAIMSObject</code>	a FAIMS object
<code>neighbourhoodSizeCol</code>	number of neighbouring columns to include in the neighbourhood
<code>neighbourhoodSizeRow</code>	number of neighbouring rows to include in the neighbourhood
<code>alpha</code>	p-value required for a pixel to be considered correlated to its neighbours
<code>plot</code>	logical - display plots with useful information?

Value

A FAIMS data matrix

`evidence_for_k` *Calculate the Evidence for k*

Description

Calculate the evidence for retaining k PCs in a PCA analysis, as in Minka 2000, "Automatic Choice of dimensionality for PCA"

Usage

```
evidence_for_k(prcompObj, k)
```

Arguments

```
prcompObj    a prcomp object  
k            k to assess
```

Value

log evidence for retaining k PCs

FeatureSelection *perform a feature selection using a Wilcoxon rank-sum test*

Description

perform a feature selection using a Wilcoxon rank-sum test

Usage

```
FeatureSelection(dataMatrix, targetValues)
```

Arguments

```
dataMatrix    a numeric matrix of training data  
targetValues  class labels (must be a logical vector)
```

Value

a list of scores, one for each sample. Lower is better.

`findNoiseFaimsData.localCorr`

Remove background noise from FAIMS data using local correlations

Description

The signal in FAIMS data exhibits a high degree of local correlation. This function zeroes out a fraction of pixels with the lowest degree of local correlation.

Usage

```
findNoiseFaimsData.localCorr(FAIMSObject, neighbourhoodSizeCol = 1,  
neighbourhoodSizeRow = 1, fractionToRemove = 0.65, plot = TRUE)
```

Arguments

`FAIMSObject` a FAIMS object

`neighbourhoodSize`
the size of the neighbourhood in which to look for local correlations

`alpha` p-value required for a pixel to be considered correlated to its neighbours

Value

Logical vector containing TRUE when columns are noise

`findNoiseFaimsData.sd`

Identify background noise from FAIMS data

Description

Identifies pixels likely to be solely noise based on standard deviation.

Usage

```
findNoiseFaimsData.sd(faimsObject, fractionNoise = 0.65, plot = TRUE)
```

Arguments

`fractionNoise` The percentage of pixels to identify as "noise"
`FAIMSObject` FAIMS object

Value

Logical vector containing TRUE when columns are noise

`generateFolds` *Generate a division of a data set into folds*

Description

Generate a division of a data set into folds

Usage

```
generateFolds(targetValues, nFolds, stratified = T)
```

Arguments

`targetValues` the class labels
`nFolds` number of folds
`stratified` whether folds should be stratified by `targetValues`

Value

a vector giving the fold for each sample

Examples

```
classes <- sample(c(rep(TRUE, 20), rep(FALSE, 50)), 70)  
folds <- generateFolds(classes, 10, T)
```

`getNeighbourIndices`

Get Neighbour Indices

Description

Return the vector indices of a matrix cell's neighbours

Usage

```
getNeighbourIndices(index, width, height, faimsDim)
```

Arguments

<code>index</code>	index of the cell in the vector
<code>width</code>	width of neighbourhood
<code>height</code>	height of neighbourhood
<code>faimsDim</code>	FAIMS matrix dimensions

Value

a vector of neighbour indices

`getRocCurve`

Return a ROC curve object, given input class probabilities

Description

Return a ROC curve object, given input class probabilities

Usage

```
getRocCurve(predictions, targetValues, titleString = "", ci = FALSE,  
lr = FALSE, plotCurve = FALSE)
```

Arguments

`predictions` vector of predictions
`targetValues` vector of class labels
`titleString` title string if `plotCurve==TRUE`
`ci` logical - compute confidence interval?
`lr` logical - compute likelihood ratios?
`plotCurve` logical - plot the resulting ROC curve?

Value

a `pROC::roc` object

`knnMeanDistance` *Find mean k-nearest neighbour distance for each sample*

Description

Compute mean k-nn distances for outlier detection

Usage

```
knnMeanDistance(dataMatrix, max.k = 10)
```

Arguments

`dataMatrix` a data matrix
`max.k` max k for mean k-nn distance

Value

a `nrow(dataMatrix)` by `k` matrix with the mean k-nn distance for each sample, with `k` along the columns and samples along the rows

<code>nKeep</code>	<i>Select the top n features</i>
--------------------	----------------------------------

Description

Select the top n features

Usage

```
nKeep(scores, nKeep)
```

Arguments

<code>scores</code>	a list of feature p-values
<code>nKeep</code>	the number to keep

Value

A list of feature indices

<code>plotFAIMSdata</code>	<i>Plot FAIMS data matrices</i>
----------------------------	---------------------------------

Description

Plot FAIMS data matrices

Usage

```
plotFAIMSdata(FAIMSObject, rowsToPlot, plotLayout = 1, absolute = T,  
...)
```

Arguments

<code>rowsToPlot</code>	the rows to plot
<code>plotLayout</code>	a matrix to specify the layout for multiple plots per page (default is one plot per page)
<code>absolute</code>	whether to plot the absolute value of the FAIMS data.
<code>...</code>	additional parameters to pass to <code>image</code>
<code>dataMatrix</code>	a matrix of FAIMS data

<code>plotRocCurve</code>	<i>Plot a ROC curve with CI</i>
---------------------------	---------------------------------

Description

Plot a ROC curve with CI

Usage

```
plotRocCurve(rocCurve, titleString = "", ci = TRUE)
```

Arguments

<code>rocCurve</code>	A ROC object
<code>titleString</code>	Title for plot
<code>ci</code>	Plot confidence intervals if TRUE

<code>prettyFAIMSPlot</code>	<i>Plot FAIMS data matrices with axes and ion current scale</i>
------------------------------	---

Description

Plot FAIMS data matrices with axes and ion current scale

Usage

```
prettyFAIMSPlot(FAIMSObject, rowToPlot, runToPlot = 1, title = "")
```

Arguments

<code>rowToPlot</code>	the row to plot
<code>runToPlot</code>	the run to plot
<code>dataMatrix</code>	a matrix of FAIMS data

<code>progBarInit</code>	<i>Initialise a progress bar object</i>
--------------------------	---

Description

Initialise a progress bar object

Usage

```
progBarInit(range)
```

Arguments

<code>range</code>	the range of values to cover
--------------------	------------------------------

Value

a progress bar object

<code>progBarUpdate</code>	<i>Update a progress bar object</i>
----------------------------	-------------------------------------

Description

Update a progress bar object

Usage

```
progBarUpdate(bar.obj, new.value)
```

Arguments

<code>a</code>	progress bar object
<code>the</code>	value to update the bar to

Value

an update progress bar object

`ReadInFaimsData` *Read in FAIMS data from a list of files*

Description

Read in FAIMS data from a list of files

Usage

```
ReadInFaimsData(fileList, dec = ".", minFlowRate = 1.9)
```

Arguments

`fileList` A list of files containing FAIMS data (in ASCII format)

`dec` Decimal symbol

`minFlowRate` minimum acceptable flow rate. If min flow rate falls below this threshold an error is thrown

Value

A FAIMS object

`ReadInFaimsDirectories`
Read in FAIMS files from a set of directories

Description

Search for directories containing FAIMS data below a specified directory, and return a named matrix with one row per directory. This function will check that there is at least one ingested file for each directory in `/codedataPath`, and return a warning if not. If multiple matching files are found an error will be thrown.

Usage

```
ReadInFaimsDirectories(dataPath, filePattern = ".*[.](txt|asc)",  
dec = ".", minFlowRate = 1.9)
```

Arguments

<code>dataPath</code>	the directory containing the data
<code>filePattern</code>	a regular expression. All files matching this regular expression will be included.
<code>dec</code>	Decimal symbol
<code>minFlowRate</code>	minimum acceptable flow rate. If min flow rate falls below this threshold an error is thrown

Value

A named matrix with one row per data file read

`ReadInFaimsDirectoriesAutosampler`

Read FAIMS data generated using an autosampler

Description

Read FAIMS data generated using an autosampler

Usage

```
ReadInFaimsDirectoriesAutosampler(dataPath, numRuns,  
filePattern = ".*[.](txt|asc)", runNumPattern = "[0-9]+[.](txt|asc)",  
dec = ".", minFlowRate = 1.9, dirPattern = ".*")
```

Arguments

<code>dataPath</code>	The directory to scan for data
<code>numRuns</code>	The number of runs to take
<code>filePattern</code>	a regex which should match the names of the files you want to ingest
<code>runNumPattern</code>	a regex which should match the run number of your data. Any characters found will be stripped and the result converted using <code>as.numeric</code>
<code>dec</code>	Decimal symbol
<code>minFlowRate</code>	minimum acceptable flow rate. If min flow rate falls below this threshold an error is thrown

Value

A FAIMS object

ReadInFaimsDirectoriesMultiFile

Read and combine multiple FAIMS data files per sample

Description

Search for directories containing FAIMS data below a specified directory, and return a named matrix with one row per directory. Multiple file patterns may be passed as a character vector to combine multiple files per directory.

Usage

```
ReadInFaimsDirectoriesMultiFile(dataPath,  
filePatterns = ".*[.](txt|asc)", dec = ".", minFlowRate = 1.9)
```

Arguments

<code>dataPath</code>	the directory containing the data
<code>filePatterns</code>	a character vector of regular expressions. All files matching this regular expression will be included.
<code>dec</code>	Decimal symbol
<code>minFlowRate</code>	minimum acceptable flow rate. If min flow rate falls below this threshold an error is thrown

Value

a named matrix with one row per directory

<code>runFAIMS</code>	<i>Run a standard analysis given a FAIMS object and class labels</i>
-----------------------	--

Description

Run a standard analysis given a FAIMS object and class labels

Usage

```
runFAIMS(FAIMSObject, targetValues, models = c("rf", "glmnet",  
"svmRadial", "svmLinear", "gbm", "nnet", "glm"),  
modelSelectFolds = NULL, modelSelectScores = NULL,  
bestModelFolds = NULL, bestModelScores = NULL, waveletData = NULL,  
SGoF = TRUE, nKeep = TRUE, extraData = NULL)
```

Arguments

<code>FAIMSObject</code>	a FAIMS object
<code>targetValues</code>	class labels
<code>models</code>	a list of <code>caret::train</code> models
<code>modelSelectFolds</code>	pre-generated folds for model selection
<code>modelSelectScores</code>	pre-generated scores for model selection
<code>bestModelFolds</code>	pre-generated folds for best model assessment
<code>bestModelScores</code>	pre-generated scores for best model assessment
<code>waveletData</code>	pre-computed wavelet data
<code>SGoF</code>	Select variables using sequential goodness of fit? (only for PCA analysis)
<code>nKeep</code>	Select variables using keep top N?
<code>extraData</code>	Additional data to feed to the classifier

Value

A list of results (see `out$bestModelSummary` and `out$modelSelectSummary` for a summary of results)

<code>runFold</code>	<i>Utility function to create predictions for a given fold and set of variables</i>
----------------------	---

Description

Utility function to create predictions for a given fold and set of variables

Usage

```
runFold(trainingData, trainingTargets, testData, models, tuneKFolds,  
tuneRepeats, tuneLength, keep, PCA, verbose, heatmap, extraTrainingData,  
extraTestData)
```

Arguments

<code>trainingData</code>	A data frame of training data
<code>trainingTargets</code>	A logical vector of training class labels
<code>testData</code>	A data frame of test data
<code>models</code>	A vector of <code>caret::train</code> models
<code>tuneKFolds</code>	number of folds for parameter tuning
<code>tuneRepeats</code>	number of repeats for parameter tuning
<code>tuneLength</code>	number of parameters to try when tuning
<code>keep</code>	numeric vector of column indices to keep in training and test data
<code>PCA</code>	apply PCA?
<code>verbose</code>	verbose output?
<code>heatmap</code>	plot a heatmap before training models?
<code>extraTrainingData</code>	additional training data
<code>extraTestData</code>	additional test data

Value

predictions for this fold

<code>select_k</code>	<i>Select PCA PCs to retain</i>
-----------------------	---------------------------------

Description

Select the number of PCs k to retain based on the evidence for each model. As per Minka 2000, "Automatic Choice of dimensionality for PCA".

Usage

```
select_k(prcompObj)
```

Arguments

`prcompObj` a `prcomp` object

Value

k

<code>SGoF</code>	<i>Significant Goodness of Fit</i>
-------------------	------------------------------------

Description

Select features using the Significant Goodness of Fit meta-test

Usage

```
SGoF(scores, alpha)
```

Arguments

`scores` a list of feature p-values
`alpha` the significance level for the test

Value

The selected features' indices

<code>threshold</code>	<i>Select features by threshold</i>
------------------------	-------------------------------------

Description

Select features whose score exceeds a given threshold

Usage

```
threshold(scores, threshold)
```

Arguments

<code>scores</code>	a list of feature p-values
<code>threshold</code>	p-value threshold

Value

A list of feature indices

<code>WaveletTransform</code>	<i>Perform a 1D wavelet transform on a FAIMS data matrix</i>
-------------------------------	--

Description

Perform a 1D wavelet transform on a FAIMS data matrix

Usage

```
WaveletTransform(FAIMSObject, discardTopLevels = 2)
```

Arguments

<code>FAIMSObject</code>	a FAIMS object
<code>discardTopLevels</code>	number of levels to discard as noise (0 keeps all data)

Value

a matrix of wavelet-transformed FAIMS data

`WaveletTransform_2D`

Perform a 2D wavelet transform on a FAIMS data matrix

Description

This function expects an input matrix of dimension (nDataItems * nFeatures) each row is therefore the 1D representation of the 2D data for a single item This means that this function needs to know the underlying dimensionality of the 2D FAIMS run

Usage

```
WaveletTransform_2D(FAIMSObject, cropSize = NULL,  
discardHighestNLevels = 2)
```

Arguments

`FAIMSObject` A FAIMS object
`cropSize` size to crop image to
`discardHighestNLevels`
number of levels to discard as noise

Value

A matrix of wavelet-transformed FAIMS data

Appendix D

Mathematical Background

D.1 Matrix identities and derivatives

D.1.1 The Woodbury matrix identity

The Woodbury matrix identity states:

$$(B + UCV)^{-1} = B^{-1} - B^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1} \quad (\text{D.1})$$

where

D.1.2 Proof

We will left-multiply the right-hand-side of equation D.1 by $B + UCV$ and show equality to the identity matrix.

$$\begin{aligned} & (B + UCV) \left(B^{-1} - B^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1} \right) \\ = & BB^{-1} - BB^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1} \\ & + UCVB^{-1} - UCVB^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1} \\ = & I - U(C^{-1} + VB^{-1}U)^{-1}VB^{-1} + UCVB^{-1} \\ & - UCVB^{-1}U(C^{-1} + VB^{-1}U)^{-1}VB^{-1} \\ = & I - U \left((C^{-1} + VB^{-1}U)^{-1} - C + CVB^{-1}U(C^{-1} + VB^{-1}U)^{-1} \right) VB^{-1} \\ = & I - UC \left(C^{-1}(C^{-1} + VB^{-1}U)^{-1} - I + VB^{-1}U(C^{-1} + VB^{-1}U)^{-1} \right) VB^{-1} \\ = & I - UC \left((C^{-1} + VB^{-1}U)(C^{-1} + VB^{-1}U)^{-1} - I \right) VB^{-1} \end{aligned}$$

$$\begin{aligned}
&= \mathbf{I} - \mathbf{UC}(\mathbf{I} - \mathbf{I})\mathbf{VB}^{-1} \\
&= \mathbf{I}
\end{aligned}$$

Hence $\mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VB}^{-1}\mathbf{U})^{-1}\mathbf{VB}^{-1}$ is the inverse for $\mathbf{B} + \mathbf{UCV}$ and the identity in equation D.1 holds.

D.1.3 Matrix derivatives

Derivative of the determinant of an invertible matrix:

$$\frac{\partial}{\partial x} \det(\mathbf{X}) = \det(\mathbf{X}) \operatorname{tr}\left(\mathbf{X}^{-1} \frac{\partial \mathbf{X}}{\partial x}\right) \quad (\text{D.2})$$

Derivative of the inverse of a matrix:

$$\frac{\partial}{\partial x} \mathbf{X}^{-1} = -\mathbf{X}^{-1} \frac{\partial \mathbf{X}}{\partial x} \mathbf{X}^{-1} \quad (\text{D.3})$$

D.2 Reproducing Kernel Hilbert Spaces

D.2.1 Definitions

Inner product space

A real or complex inner product space is a vector space V over $F \in \{\mathbb{R}, \mathbb{C}\}$ together with an inner product $\langle \cdot, \cdot \rangle : V \times V \rightarrow F$ which satisfies the following axioms for all $x, y, z \in V$ and all $a \in F$ [Rudin, 1987]:

1. $\langle x, y \rangle = \overline{\langle y, x \rangle}$
2. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
3. $\langle ax, y \rangle = a \langle x, y \rangle$
4. $\langle x, x \rangle \geq 0$
5. $\langle x, x \rangle = 0 \implies x = 0$

Hilbert space

An inner product space H with inner product $\langle \cdot, \cdot \rangle_H$ which is complete¹ under the norm $\|x\|_H = \langle x, x \rangle_H^{\frac{1}{2}}$ induced by its inner product is a Hilbert space [Rudin, 1987].

¹ H is complete if for every Cauchy sequence in H its limit is also contained in H .

RKHS definition (1)

A Hilbert space H of real-valued functions on an index set \mathcal{X} is a reproducing kernel Hilbert space (RKHS) iff there exists a function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ such that:

1. $K_x: x' \mapsto k(x', x) \in H \forall x \in \mathcal{X}$
2. k has the reproducing property: $\langle f, K_x \rangle_H = f(x) \forall x \in \mathcal{X}, f \in H$

This is the definition given in Rasmussen and Williams [2006], however an equivalent definition seen elsewhere (for example, Berlinet and Thomas-Agnan [2011]), uses continuity of the evaluation functional over H .

RKHS definition (2)

Let H be a Hilbert space H of real-valued functions on an index set \mathcal{X} , and let $L_x: H \rightarrow \mathbb{R}$ be the linear functional which evaluates each function in H at $x \in \mathcal{X}$.

$$L_x: f \mapsto f(x) \tag{D.4}$$

Then H is an RKHS iff L_x is continuous for all $x \in \mathcal{X}$.

D.2.2 Equivalence of definitions

To see that RKHS definition (1) implies definition (2), we note that $L_x(f) = \langle f, K_x \rangle_H$ and show that L_x is continuous. By the Cauchy-Schwarz inequality and linearity of the inner product in the first argument,

$$\begin{aligned} L_x(f_1) - L_x(f_2) &= |\langle f_1, K_x \rangle_H - \langle f_2, K_x \rangle_H| \\ &= |\langle f_1 - f_2, K_x \rangle_H| \\ &\leq \|f_1 - f_2\|_H \|K_x\|_H \end{aligned} \tag{D.5}$$

and so L_x is uniformly continuous.

For (2) \implies (1), we need a version of the Riesz representation theorem.

Riesz representation

If L is a continuous linear functional on H , then there is a unique $y \in H$ such that

$$L(x) = \langle x, y \rangle_H \quad \forall x \in H \tag{D.6}$$

Proof [Rudin, 1987]: If L is identically zero, then take $y = 0$. Otherwise, define

$$M = \{x \mid L(x) = 0\} \quad (\text{D.7})$$

By linearity of L , M is a subspace of H , since for $x_1, x_2 \in M$, $L(\alpha_1 x_1 + \alpha_2 x_2) = \alpha_1 L(x_1) + \alpha_2 L(x_2) = 0$. By continuity of L , M is complete, which can be proved easily by contradiction. Assume there is a Cauchy sequence $(x_i)_{i \in \mathbb{N}}$ in M which converges to $x \notin M$. Therefore, $L(x) \neq 0$, however by continuity of L

$$\begin{aligned} L(x) &= \lim_{i \rightarrow \infty} L(x_i) \\ &= \lim_{i \rightarrow \infty} 0 \\ &= 0 \end{aligned} \quad (\text{D.8})$$

A contradiction. Hence, M is complete. Since $L(x) \neq 0$ for some $x \in H$, $M^\perp = \{x \in H \mid \langle x, y \rangle_H = 0 \ \forall y \in M\}$ does not only contain zero. Hence, there exists $z \in M^\perp$ with $\|z\| = 1$. Define

$$u = L(x)z - L(z)x \quad (\text{D.9})$$

$L(u) = L(x)L(z) - L(z)L(x) = 0$, so $u \in M$. Therefore $\langle u, z \rangle_H = 0$, and we have:

$$0 = \langle u, z \rangle_H \quad (\text{D.10})$$

$$= \langle L(x)z - L(z)x, z \rangle_H \quad (\text{D.11})$$

$$= L(x) \langle z, z \rangle_H - L(z) \langle x, z \rangle_H \quad (\text{D.12})$$

$$= L(x) - L(z) \langle x, z \rangle_H \quad (\text{D.13})$$

$$\implies L(x) = L(z) \langle x, z \rangle_H \quad (\text{D.14})$$

Hence setting $y = \bar{\alpha}z$ where $\alpha = L(z)$, we have $L(x) = \langle x, y \rangle \ \forall x \in H$.

For uniqueness, consider that if y, y' satisfy equation D.6, then $\langle x, y \rangle_H = \langle x, y' \rangle_H$ for all $x \in H$, so $\langle x, y - y' \rangle_H = 0$ for all $x \in H$, and specifically $\langle y - y', y - y' \rangle_H = 0$, and hence $y = y'$.

RKHS definition equivalence continued

Now, we can approach definition (2) \implies (1). Since L_x is continuous, the Riesz representation theorem tells us that there exists a unique $K_x \in H$ such that $L_x(f) = \langle f, K_x \rangle_H$ for all $f \in H$. Then setting $k(x, x') = K_x(x')$, we have a k which satisfies the first definition of an RKHS.

With this equivalence established, we can note an interesting fact about the RKHS, namely that since the evaluation functional L_x is continuous for all $x \in \mathcal{X}$, if two functions $f, g \in H$ are close in the norm induced by $\langle \cdot, \cdot \rangle_H$ then $f(x)$ and $g(x)$ are also close in value for all $x \in \mathcal{X}$. More formally, if $\lim_{n \rightarrow \infty} \|f_n - f\|_H = 0$, then $\lim_{n \rightarrow \infty} f_n(x) = f(x) \forall x \in \mathcal{X}$, since for any $x \in \mathcal{X}$

$$\begin{aligned} |f_n(x) - f(x)| &= |\langle f_n, K_x \rangle_H - \langle f, K_x \rangle_H| \\ &= |\langle f_n - f, K_x \rangle_H| \\ &\leq \|f_n - f\|_H \|K_x\|_H \end{aligned}$$

D.2.3 The Moore–Aronszajn theorem

The Moore-Aronszajn theorem provides the link between a positive definite kernel function and the RKHS which it implicitly projects into.

Moore-Aronszajn theorem

Let $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ be positive definite. Then there is a unique RKHS H for which k is the reproducing kernel.

Sketch of the proof

We present here a brief outline of the proof of the Moore-Aronszajn theorem as presented in Sejdinovic and Gretton [2012]. First, we define the pre-RKHS H_0 to be the linear span of the set $\{K_x \mid x \in \mathcal{X}\}$, and define an inner product

$$\langle f, g \rangle_{H_0} = \sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j k(x_i, y_j) \quad (\text{D.15})$$

where $f: x \mapsto \sum_{i=1}^n \alpha_i k(x, x_i)$ and $g: x \mapsto \sum_{j=1}^m \beta_j k(x, y_j)$.

First, we show that $\langle \cdot, \cdot \rangle_{H_0}$ is a valid inner product on H_0 . It is independent of the specific values of α_i and β_j chosen to represent f and g :

$$\langle f, g \rangle_{H_0} = \sum_{i=1}^n \alpha_i g(x_i) = \sum_{j=1}^m \beta_j f(y_j) \quad (\text{D.16})$$

It is trivial to show that $f = 0 \implies \langle f, f \rangle_{H_0} = 0$. To show the reverse, assume $\langle f, f \rangle_{H_0} = 0$, let $f(x) = \sum_{i=1}^n \alpha_i k(x, x_i)$, and set $a_i = -k(x, x) \alpha_i$, $i = 1 \dots n$,

$a_{n+1} = f(x)$, $x_{n+1} = x$. Since k is positive definite we have

$$\begin{aligned} 0 &\leq \sum_{i=1}^{n+1} \sum_{j=1}^{n+1} a_i a_j k(x_i, x_j) \\ &= a^2 \langle f, f \rangle_{H_0} + 2a |f(x)|^2 + |f(x)|^2 k(x, x) \end{aligned}$$

Since this inequality holds for all a and the r.h.s. is a quadratic function, we have that $|f(x)|^4 \leq |f(x)|^2 k(x, x) \langle f, f \rangle_{H_0}$ and hence $f(x) = 0$ for all x .

To show that H_0 is a pre-RKHS, we must demonstrate that:

1. The evaluation functions L_x on H_0 are continuous.
2. Any Cauchy sequence $(f_n)_{n \in \mathbb{N}}$ in H_0 which converges pointwise to 0 also converges to 0 in H_0 -norm.

With H_0 in place, we define H to be

$$H = \{f \mid \exists (f_n)_{n \in \mathbb{N}}, f_n \in H_0, \lim_{n \rightarrow \infty} f_n(x) = f(x) \forall x \in \mathcal{X}\} \quad (\text{D.17})$$

and given Cauchy sequences $(f_n), (g_n)$ in H_0 converging to f, g in H , we define $\langle f, g \rangle_H$ as:

$$\langle f, g \rangle_H = \lim_{n \rightarrow \infty} \langle f_n, g_n \rangle_{H_0} \quad (\text{D.18})$$

We now need to prove that H is an RKHS. To do this, we can consider that the two properties above hold if and only if:

1. $H_0 \subset H \subset \mathbb{R}^{\mathcal{X}}$, and the topology induced in H_0 by $\langle \cdot, \cdot \rangle_{H_0}$ is the same as the topology induced on H_0 by H .
2. H has a reproducing kernel $k(x, y)$.

Proving that these conditions imply that H_0 is a pre-RKHS is straightforward:

1. If H has a reproducing kernel then L_x is continuous on H , and therefore continuous on H_0 since the topology induced by H on H_0 matches the topology induced by $\langle \cdot, \cdot \rangle_{H_0}$.
2. Let $(f_n)_{n \in \mathbb{N}}$ be a Cauchy sequence in $H_0 \subset H$ which converges point-wise to 0, and let $f \in H$ be the limit of the sequence, that is $\lim_{n \rightarrow \infty} \|f_n - f\|_H = 0$. Then by the Cauchy-Schwarz inequality, $f_n(x) - f(x) = \langle f_n - f, K_x \rangle_H \leq \|f_n - f\|_H \|K_x\|_H \rightarrow 0$ as $n \rightarrow \infty$. Hence, $f(x) = \lim_{n \rightarrow \infty} f_n(x) - f(x) = 0 \forall x \in \mathcal{X}$, and $f = 0$.

Proving the reverse is more involved, and the steps are only outlined here. We must show that:

1. The inner product is well-defined and independent of the choice of (f_n) and (g_n) .
2. The requirements of an inner product space are met, particularly that $f = 0 \iff \langle f, f \rangle_H = 0$.
3. The evaluation functionals L_x are continuous on H .
4. H is complete.

These are attacked as follows:

1. Convergence is demonstrated by applying the Cauchy-Schwarz inequality to $|\langle f_n, g_n \rangle - \langle f_m, g_m \rangle|$ and thereby showing $(\langle f_n, g_n \rangle)_{n \in \mathbb{N}}$ is a Cauchy sequence in \mathbb{R} . Independence from representation is shown by again applying the Cauchy-Schwarz inequality, this time to $|\langle f_n, g_n \rangle - \langle f'_n, g'_n \rangle|$, where $(f'_n), (g'_n)$ are alternative Cauchy sequences converging to f and g respectively.
2. $f = 0 \implies \langle f, f \rangle = 0$ is trivial. For the reverse, we use the continuity of the linear operators L_x on H_0 , which implies that L_x is bounded: $f(x) = \lim_{n \rightarrow \infty} f_n(x) = \lim_{n \rightarrow \infty} L_x(f_n) \leq \lim_{n \rightarrow \infty} \|L_x\| \|f_n\| = 0$

3. To show the evaluation functionals are continuous on H , first we show that H_0 is dense in H , that is, that point-wise convergence of a Cauchy sequence in H_0 implies convergence in H -norm, which follows straightforwardly from taking the definition of a Cauchy sequence ($\forall \epsilon > 0 \exists N \in \mathbb{N}$ s.t. $\|f_n - f_m\| < \epsilon \forall m, n \geq N$), fixing $n = N$ and letting m tend to infinity, to give from the definition of the inner product $\|f - f_N\|_H^2 = \lim_{m \rightarrow \infty} \|f_m - f_N\|_{H_0}^2 \leq \epsilon^2$, and hence (f_n) converges to f in $\|\cdot\|_H$.

We now prove the evaluation functionals are continuous at 0, and by linearity this extends to the whole of H . First, by the properties of a pre-RKHS L_x is continuous at 0 in H_0 . We then use the fact that H_0 is dense in H to find for $f \in H$ close to 0 a $g \in H_0$ close to f in H -norm for which $f(x)$ is also close to $g(x)$. We then use f and g to show that $\|f\|_H < \nu \implies |f(x)| < \epsilon$ and hence L_x is continuous in H .

4. Completeness of H is shown by using the fact that H_0 is dense in H : for a Cauchy sequence (f_n) in H , construct a parallel Cauchy sequence (g_n) in H_0 such that $\|f_n - g_n\|_H < 1/n$. This converges point-wise to the same function,

which can be shown using linearity of the evaluation functionals. Finally, show that the point-wise limit f is also the limit in H -norm by applying the triangle inequality to f_n, g_n and f .

Since H is complete, using the Cauchy convergence criterion we can show that it is equivalent to the set of all convergent series

$$\sum_{n=1}^{\infty} a_n, \quad a_n \in H_0 \tag{D.19}$$

The partial sums of such a convergent series form a Cauchy sequence in H_0 , so the limit of the series is therefore in H . Conversely for every element $b \in H$ there exists by denseness of H_0 in H a Cauchy sequence $(b_n)_{n \in \mathbb{N}}$ with $b_n \in H_0$ such that $\lim_{n \rightarrow \infty} b_n = b$. By defining $a_1 = b_1$ and $a_n = b_n - a_{n-1}$ for $n > 1$, we have $b_n = \sum_{i=1}^n a_i$ and can therefore write this as the convergent series $\sum_{i=1}^{\infty} a_i = b$.

Now, since H_0 is defined to be the linear span of $\{K_x \mid x \in \mathcal{X}\}$, we can re-write a_n as $a_n = \sum_{i=1}^m \alpha_i K_{x_{ni}}$ where $\alpha_i \in \mathbb{R}$, and therefore see that any element $a = \sum_{n=1}^{\infty} a_n \in H$ can be written as an infinite sum of instances of the kernel function k instantiated at points in \mathcal{X} . Since the RKHS associated with a kernel function is unique, this characterises the feature space of functions which GPs operate in.

Bibliography

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- Andreas Adler, Sebastian Geiger, Anne Keil, Harald Bias, Philipp Schatz, Jens Dhein, Mathias Zimmermann, Rudolf Tauber, Bertram Wiedenmann, et al. Improving compliance to colorectal cancer screening using blood and stool based tests in patients refusing screening colonoscopy in Germany. *BMC Gastroenterology*, 14(1):183, 2014.
- Catherine Alix-Panabières, Heidi Schwarzenbach, and Klaus Pantel. Circulating tumor cells and circulating tumor DNA. *Annual Review of Medicine*, 63:199–215, 2012.
- Mònica Alonso and Juan M Sanchez. Analytical challenges in breath analysis and its application to exposure monitoring. *TrAC Trends in Analytical Chemistry*, 44:78–89, 2013.
- Anton Amann, Ben de Lacy Costello, Wolfram Miekisch, Jochen Schubert, Bogusław Buszewski, Joachim Pleil, Norman Ratcliffe, and Terence Risby. The human volatilome: volatile organic compounds (VOCs) in exhaled breath, skin emanations, urine, feces and saliva. *Journal of Breath Research*, 8(3):034001, 2014.
- Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W Hogg, and Michael O’Neil. Fast direct methods for gaussian processes. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):252–265, 2015.
- Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008. ISSN 1063-6919. doi: 10.1109/CVPR.2008.4587583.

- Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–27, Helsinki, Finland, 2002. Springer.
- Ramesh P Arasaradnam, Michael J McFarlane, Courtenay Ryan-Fisher, Erik Westenbrink, Paula Hodges, Matthew G Thomas, Samantha Chambers, Nicola O’Connell, Catherine Bailey, Christopher Harmston, et al. Detection of colorectal cancer (CRC) by urinary volatile organic compound analysis. *PLoS One*, 9(9):e108750, 2014.
- Amel Bajtarevic, Clemens Ager, Martin Pienz, Martin Klieber, Konrad Schwarz, Magdalena Ligor, Tomasz Ligor, Wojciech Filipiak, Hubert Denz, Michael Fiegl, et al. Noninvasive detection of lung cancer by analysis of exhaled breath. *BMC Cancer*, 9(1):348, 2009.
- Henrik Bengtsson. *R.cache: Fast and Light-Weight Caching (Memoization) of Objects and Results to Speed Up Computations*, 2018. URL <https://CRAN.R-project.org/package=R.cache>. R package version 0.13.0.
- James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.
- James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, Atlanta, USA, 2013. JMLR.
- James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, Granada, Spain, 2011.
- Alain Berlinet and Christine Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer Science & Business Media, 2011.
- Myron G Best, Nik Sol, Irsan Kooi, Jihane Tannous, Bart A Westerman, François Rustenburg, Pepijn Schellen, Heleen Verschueren, Edward Post, Jan Koster, et al. RNA-Seq of tumor-educated platelets enables blood-based pan-cancer, multiclass, and molecular pathway cancer diagnostics. *Cancer Cell*, 28(5):666–676, 2015.
- Chetan Bettgowda, Mark Sausen, Rebecca J Leary, Isaac Kinde, Yuxuan Wang, Nishant Agrawal, Bjarne R Bartlett, Hao Wang, Brandon Lubber, Rhoda M

- Alani, et al. Detection of circulating tumor DNA in early-and late-stage human malignancies. *Science Translational Medicine*, 6(224):224ra24–224ra24, 2014.
- Christopher M Bishop. Bayesian PCA. In *Advances in Neural Information Processing Systems*, volume 11, pages 382–388, Cambridge, Mass., USA, 1999. MIT Press.
- Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer Science & Business Media, 2006.
- Ramin Bostanabad, Tucker Kearney, Siyu Tao, Daniel W Apley, and Wei Chen. Leveraging the nugget parameter for efficient gaussian process modeling. *International Journal for Numerical Methods in Engineering*, 114(5):501–516, 2018.
- Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pages 177–186. Springer, Paris, France, 2010.
- Matthieu Bray, Esther Koller-Meier, Pascal Müller, Luc Van Gool, and Nicol N Schraudolph. 3D hand tracking by rapid stochastic gradient descent using a skinning model. In *In 1st European Conference on Visual Media Production (CVMP, London, UK, 2004a*. Citeseer.
- Matthieu Bray, Esther Koller-Meier, Nicol N Schraudolph, and Luc Van Gool. Stochastic meta-descent for tracking articulated structures. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW’04.*, pages 7–7, Washington DC, USA, 2004b. IEEE.
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- Leo Breiman. *Classification and Regression Trees*. Routledge, 2017.
- Bogusław Buszewski, Tomasz Ligor, Tadeusz Jezierski, Anna Wenda-Piesik, Marta Walczak, and Joanna Rudnicka. Identification of volatile lung cancer markers by gas chromatography–mass spectrometry: comparison with discrimination by canines. *Analytical and Bioanalytical Chemistry*, 404(1):141–146, 2012.
- Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.
- Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery*, 30(4):891–927, 2016.

- Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168, Pittsburgh, Pa., USA, 2006. ACM.
- Antonio Carvajal-Rodríguez, Jacobo de Uña-Alvarez, and Emilio Rolán-Alvarez. A new multitest correction (SGoF) that increases its statistical power when increasing the number of tests. *BMC Bioinformatics*, 10(1):209, 2009.
- Raymond B Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.
- Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.
- Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco, Calif., USA, 2016. ACM.
- Joshua D Cohen, Lu Li, Yuxuan Wang, Christopher Thoburn, Bahman Afsari, Ludmila Danilova, Christopher Douville, Ammar A Javed, Fay Wong, Austin Mattox, et al. Detection and localization of surgically resectable cancers with a multi-analyte blood test. *Science*, 359(20):926–930, 2018.
- Jean-Nicolas Cornu, Géraldine Cancel-Tassin, Valérie Ondet, Caroline Girardet, and Olivier Cussenot. Olfactory detection of prostate cancer by dogs sniffing urine: a step forward in early diagnosis. *European Urology*, 59(2):197–201, 2011.
- Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- F Costa, MG Mumolo, M al Bellini, MR Romano, L Ceccarelli, P Arpe, C Sterpi, S Marchi, and G Maltinti. Role of faecal calprotectin as non-invasive marker of intestinal inflammation. *Digestive and Liver Disease*, 35(9):642–647, 2003.
- Council of the European Union. Directive 2004/42/CE of the European Parliament and of the Council of 21 April 2004 on the limitation of emissions of volatile organic compounds due to the use of organic solvents in certain paints and varnishes and vehicle refinishing products and amending Directive 1999/13/EC, 2004.

- JA Covington, MP van der Schee, ASL Edge, B Boyle, RS Savage, and RP Arasaradnam. The application of FAIMS gas analysis in medical diagnostics. *Analyst*, 140(20):6775–6781, 2015.
- James A Covington, Eric W Westenbrink, Nathalie Ouaret, Ruth Harbord, Catherine Bailey, Nicola O’Connell, James Cullis, Nigel Williams, Chuka U Nwokolo, Karna D Bardhan, et al. Application of a novel tool for diagnosing bile acid diarrhoea. *Sensors*, 13(9):11899–11912, 2013.
- Ian A Cree. Liquid biopsy for cancer patients: principles and practice. *Pathogenesis*, 2(1–2):1–4, 2015.
- Garrett M Dancik and Karin S Dorman. mlegp: statistical analysis for computer models of biological systems using R. *Bioinformatics*, 24(17):1966, 2008.
- Christian Darken, Joseph Chang, and John Moody. Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 3–12, Denver, Colo., USA, 1992. IEEE.
- Abhirup Datta, Sudipto Banerjee, Andrew O Finley, and Alan E Gelfand. Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.
- Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941, Montreal, Quebec, Canada, 2014.
- Alex Davies and Zoubin Ghahramani. The random forest kernel and other kernels for big data from random partitions. *arXiv preprint arXiv:1402.4293*, 2014.
- Roger S Day. Planning clinically relevant biomarker validation studies using the “number needed to treat” concept. *Journal of Translational Medicine*, 14(1):117, 2016.
- M Di Lena, F Porcelli, and DF Altomare. Volatile organic compounds as new biomarkers for colorectal cancer: a review. *Colorectal Disease*, 18(7):654–663, 2016.
- Thomas J DiCiccio and Bradley Efron. Bootstrap confidence intervals. *Statistical Science*, pages 189–212, 1996.

- Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.
- Sarah J Dixon, Yun Xu, Richard G Brereton, Helena A Soini, Milos V Novotny, Elisabeth Oberzaucher, Karl Grammer, and Dustin J Penn. Pattern recognition of gas chromatography mass spectrometry of human volatiles in sweat to distinguish the sex of subjects and determine potential discriminatory marker peaks. *Chemometrics and Intelligent Laboratory Systems*, 87(2):161–172, 2007.
- John H Duffus, Monica Nordberg, and Douglas M Templeton. Glossary of terms used in toxicology, (IUPAC Recommendations 2007). *Pure and Applied Chemistry*, 79(7):1153–1344, 2007.
- Dirk Eddelbuettel and James Joseph Balamuta. Extending R with C++: A Brief Introduction to Rcpp. *PeerJ Preprints*, 5:e3188v1, aug 2017. ISSN 2167-9843. doi: 10.7287/peerj.preprints.3188v1. URL <https://doi.org/10.7287/peerj.preprints.3188v1>.
- Katharina Eggenberger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, page 3, Montreal, Quebec, Canada, 2013.
- Collin B Erickson, Bruce E Ankenman, and Susan M Sanchez. Comparison of Gaussian process modelling software. *European Journal of Operational Research*, 266(1):179–192, 2018.
- Siavash Esfahani, Nidhi M Sagar, Ioannis Kyrou, Ella Mozdiak, Nicola O’Connell, Chuka Nwokolo, Karna D Bardhan, Ramesh P Arasaradnam, and James A Covington. Variation in gas and volatile compound emissions from human urine as it ages, measured by an electronic nose. *Biosensors*, 6(1):4, 2016.
- Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.
- National Institute for Health and Care Excellence (2013). Faecal calprotectin diagnostic tests for inflammatory diseases of the bowel. NICE guideline (DG11), 2013.
- National Institute for Health and Care Excellence (2015). Suspected cancer: recognition and referral. NICE guideline (NG12), 2015.

- National Institute for Health and Care Excellence (2017). Quantitative faecal immunochemical tests to guide referral for colorectal cancer in primary care. NICE guideline (DG30), 2017.
- Daniel Foreman-Mackey, Eric Agol, Sivaram Ambikasaran, and Ruth Angus. Fast and scalable gaussian process modeling with applications to astronomical time series. *The Astronomical Journal*, 154(6):220, 2017.
- George Forman and Martin Scholz. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010.
- Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL <http://www.jstatsoft.org/v33/i01/>.
- Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.
- Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.
- Simon Garnier. *viridis: Default Color Maps from 'matplotlib'*, 2018. URL <https://CRAN.R-project.org/package=viridis>. R package version 0.5.1.
- Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- Paul Gilbert and Ravi Varadhan. *numDeriv: Accurate Numerical Derivatives*, 2016. URL <https://CRAN.R-project.org/package=numDeriv>. R package version 2016.8-1.
- Carl Gold and Peter Sollich. Model selection for support vector machine classification. *Neurocomputing*, 55(1-2):221–249, 2003.
- John C Gower. Generalized Procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.
- GRAIL Inc. Publications. <https://grail.com/science/publications/>, January 2019. Date accessed: 20th January 2019.
- Robert B. Gramacy. laGP: Large-scale spatial modeling via local approximate Gaussian processes in R. *Journal of Statistical Software*, 72(1):1–46, 2016. doi: 10.18637/jss.v072.i01.

- Robert B. Gramacy and Daniel W. Apley. Local gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578, 2015. doi: 10.1080/10618600.2014.914442. URL <https://doi.org/10.1080/10618600.2014.914442>.
- Rajarshi Guhaniyogi and Sudipto Banerjee. Meta-kriging: Scalable bayesian modeling and inference for massive spatial datasets. *Technometrics*, 60(4):430–444, 2018.
- Yosuke Hanai, Ken Shimono, Koichi Matsumura, Anil Vachani, Steven Albelda, Kunio Yamazaki, Gary K Beauchamp, and Hiroaki Oka. Urinary volatile compounds as biomarkers for lung cancer. *Bioscience, Biotechnology, and Biochemistry*, 76(4):679–684, 2012.
- Matthew J. Heaton, Abhirup Datta, Andrew O. Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, Robert B. Gramacy, Dorit Hammerling, Matthias Katzfuss, Finn Lindgren, Douglas W. Nychka, Furong Sun, and Andrew Zammit-Mangion. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, Dec 2018. ISSN 1537-2693. doi: 10.1007/s13253-018-00348-w. URL <https://doi.org/10.1007/s13253-018-00348-w>.
- Ralf Herbrich, Neil D Lawrence, and Matthias Seeger. Fast sparse Gaussian process methods: The informative vector machine. In *Advances in neural information processing systems*, pages 625–632, Vancouver, Canada, 2003.
- Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The Annals of Statistics*, pages 1171–1220, 2008.
- John L Horn. A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2):179–185, 1965.
- Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.
- Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.
- Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

- Maik A Jochmann, Magda P Kmiecik, and Torsten C Schmidt. Solid-phase dynamic extraction for the enrichment of polar volatile organic compounds from water. *Journal of Chromatography A*, 1115(1-2):208–216, 2006.
- Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL <http://www.jstatsoft.org/v11/i09/>.
- J Karnon, J Peters, J Platt, J Chilcott, E McGoogan, and N Brewer. Liquid-based cytology in cervical screening: an updated rapid and systematic review and economic analysis. *Health Technology Assessment*, 8(20), 2004.
- Matthias Katzfuss. A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association*, 112(517):201–214, 2017.
- C. D. Keeling and T. P. Whorf. Scripps Institution of Oceanography (SIO), University of California, La Jolla, California USA 92093-0220 <ftp://cdiac.esd.ornl.gov/pub/maunaloa-co2/maunaloa.co2>, via the R datasets package, 2018.
- Brian W Kernighan and Dennis M Ritchie. *The C Programming Language*. Prentice Hall, second edition, 1988.
- Hyunjik Kim and Yee Whye Teh. Scaling up the Automatic Statistician: Scalable structure discovery using Gaussian processes. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 575–584. PLMR, 2018.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI’95 Proceedings of the 14th International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1145, Montreal, Canada, 1995.
- Beata M Kolakowski and Zoltán Mester. Review of applications of high-field asymmetric waveform ion mobility spectrometry (FAIMS) and differential mobility spectrometry (DMS). *Analyst*, 132(9):842–864, 2007.
- Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

- Wojtek J Krzanowski and David J Hand. *ROC Curves for Continuous Data*. Chapman & Hall/CRC - Taylor & Francis Group, 2009.
- Max Kuhn, Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt. *caret: Classification and Regression Training*, 2018. URL <https://CRAN.R-project.org/package=caret>. R package version 6.0-80.
- S. Y. Kung. *Kernel Methods and Machine Learning*. Cambridge University Press, 2014.
- Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6(Oct): 1679–1704, 2005.
- Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*, pages 329–336, Vancouver, British Columbia, Canada, 2004.
- Neil D Lawrence and Joaquin Quiñero-Candela. Local distance preservation in the GP-LVM through back constraints. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 513–520, Pittsburgh, Pennsylvania, USA, 2006. ACM.
- Andy Liaw and Matthew Wiener. Classification and regression by randomForest. *R News*, 2(3):18–22, 2002. URL <https://CRAN.R-project.org/doc/Rnews/>.
- Blake MacDonald, Pritam Ranjan, and Hugh Chipman. GPfit: An R package for fitting a Gaussian process model to deterministic simulator outputs. *Journal of Statistical Software*, 64(12):1–23, 2015. URL <http://www.jstatsoft.org/v64/i12/>.
- Roberto F Machado, Daniel Laskowski, Olivia Deffenderfer, Timothy Burch, Shuo Zheng, Peter J Mazzone, Tarek Mekhail, Constance Jennings, James K Stoller, Jacqueline Pyle, et al. Detection of lung cancer by sensor array analyses of exhaled breath. *American Journal of Respiratory and Critical Care Medicine*, 171(11): 1286–1291, 2005.
- David JC MacKay et al. Bayesian nonlinear modelling for the prediction competition. *ASHRAE Transactions*, 100(2):1053–1062, 1994.

- Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, pages 50–60, 1947.
- Alexander G de G Matthews, Mark van der Wilk, Tom Nickson, Keisuke Fujii, Alexis Boukouvalas, Pablo León-Villagr a, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.
- Peter J Mazzone, Jeffrey Hammel, Raed Dweik, Jie Na, Carmen Czich, Daniel Laskowski, and Tarek Mekhail. Lung cancer diagnosis by the analysis of exhaled breath with a colorimetric sensor array. *Thorax*, 2007.
- S Elizabeth McGregor, Robert J Hilsden, Feng X Li, Heather E Bryant, and Alison Murray. Low uptake of colorectal cancer screening 3 yr after release of national recommendations for screening. *The American Journal of Gastroenterology*, 102(8):1727–1735, 2007.
- Thomas P Minka. Automatic choice of dimensionality for PCA. In *Advances in Neural Information Processing Systems*, pages 598–604, Vancouver, British Columbia, Canada, 2001.
- Martin Fodslette M oller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.
- Craig Mowat, Jayne Digby, Judith A Strachan, Robyn Wilson, Francis A Carey, Callum G Fraser, and Robert JC Steele. Faecal haemoglobin and faecal calprotectin as indicators of bowel disease in patients presenting to primary care with bowel symptoms. *Gut*, 65(9):1463–1469, 2016.
- John C. Nash. On best practice optimization methods in R. *Journal of Statistical Software*, 60(2):1–14, 2014. URL <http://www.jstatsoft.org/v60/i02/>.
- Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.
- Andrew A Neath and Joseph E Cavanaugh. The Bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203, 2012.
- NHS England. Bowel cancer: overview. <https://www.nhs.uk/conditions/bowel-cancer/>, October 2016. Date accessed: 2nd December 2018.

NHS England. Home-testing kits that detect bowel cancer could almost halve invasive examinations by 2020. <https://www.england.nhs.uk/2017/09/home-testing-kits-that-detect-bowel-cancer-could-almost-halve-invasive-examinations-by-2020/>, September 2017. Date accessed: 2nd December 2018.

Naoyo Nishida, Hirohisa Yano, Takashi Nishida, Toshiharu Kamura, and Masamichi Kojiro. Angiogenesis in cancer. *Vascular Health and Risk Management*, 2(3):213, 2006.

Jamie R Nuñez, Christopher R Anderton, and Ryan S Renslow. Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PloS One*, 13(7):e0199239, 2018.

Douglas Nychka, Soutir Bandyopadhyay, Dorit Hammerling, Finn Lindgren, and Stephan Sain. A multiresolution gaussian process model for the analysis of large spatial datasets. *Journal of Computational and Graphical Statistics*, 24(2):579–599, 2015. doi: 10.1080/10618600.2014.914946. URL <https://doi.org/10.1080/10618600.2014.914946>.

Department of Health. The NHS cancer plan. http://webarchive.nationalarchives.gov.uk/20130222181549/http://www.dh.gov.uk/prod_consum_dh/groups/dh_digitalassets/@dh/@en/documents/digitalasset/dh_4014513.pdf, 2000.

Office for National Statistics. Cancer registration statistics, England: 2016, 2016.

Object Management Group (OMG). Unified Modelling Language (UML) specification, version 2.5.1. OMG File ID formal/2006-01-01 (<https://www.omg.org/spec/UML/2.5.1/PDF>), 2017.

Margaret E O’Hara, Tom H Clutton-Brock, Stuart Green, Shane O’Hehir, and Chris A Mayhew. Mass spectrometric investigations to obtain the first direct comparisons of endogenous breath and blood volatile organic compound concentrations in healthy volunteers. *International Journal of Mass Spectrometry*, 281(1-2):92–96, 2009.

Christopher J Paciorek, Benjamin Lipshitz, Wei Zhuo, Cari G Kaufman, Rollin C Thomas, et al. Parallelizing gaussian process calculations in r. *arXiv preprint arXiv:1305.4886*, 2013.

- CK Palmer, MC Thomas, C Von Wagner, and R Raine. Reasons for non-uptake and subsequent participation in the NHS bowel cancer screening programme: a qualitative study. *British Journal of Cancer*, 110(7):1705, 2014.
- Barak A Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.
- Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.
- Gang Peng, Meggie Hakim, Yoav Y Broza, Salem Billan, Roxolyana Abdah-Bortnyak, Abraham Kuten, Ulrike Tisch, and Hossam Haick. Detection of lung, breast, colorectal, and prostate cancers from exhaled breath using a single array of nanosensors. *British Journal of Cancer*, 103(4):542, 2010.
- Julian Peto, Clare Gilham, Olivia Fletcher, and Fiona E Matthews. The cervical cancer epidemic that screening has prevented in the UK. *The Lancet*, 364(9430):249–256, 2004.
- Paul DP Pharoah, Bernadette Sewell, Deborah Fitzsimmons, Hayley S Bennett, and Nora Pashayan. Cost effectiveness of the NHS breast screening programme: life table model. *BMJ*, 346:f2618, 2013.
- Michael Phillips, Kevin Gleeson, J Michael B Hughes, Joel Greenberg, Renee N Cataneo, Leigh Baker, and W Patrick McVay. Volatile organic compounds in breath as markers of lung cancer: a cross-sectional study. *The Lancet*, 353(9168):1930–1933, 1999.
- Victor D Plat, Nora van Gaal, James A Covington, Matthew Neal, Tim GJ de Meij, Donald L van der Peet, Babs Zonderhuis, Geert Kazemier, Nanne KH de Boer, and Freek Daams. Non-invasive detection of anastomotic leakage following esophageal and pancreatic surgery by urinary analysis. *Digestive Surgery*, pages 1–8, 2018.
- Martyn Plummer. *rjags: Bayesian Graphical Models using MCMC*, 2016. URL <https://CRAN.R-project.org/package=rjags>. R package version 4-6.
- Martyn Plummer et al. JAGS: A program for analysis of Bayesian graphical models using gibbs sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, volume 124, Vienna, Austria, 2003. Austrian Association for Statistical Computing (AASC) and the R Foundation for Statistical Computing.

- Philipp Probst, Bernd Bischl, and Anne-Laure Boulesteix. Tunability: Importance of hyperparameters of machine learning algorithms. *arXiv preprint arXiv:1802.09596*, 2018.
- Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.
- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018a. URL <https://www.R-project.org/>.
- R Core Team. R language definition, 07 2018b. URL <https://cran.r-project.org/doc/manuals/r-release/R-lang.html>.
- David F Ransohoff. The process to discover and develop biomarkers for cancer: a work in progress. *JNCI: Journal of the National Cancer Institute*, 100(20):1419–1420, 2008.
- Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006. ISBN 0-262-18253-X. URL <http://www.gaussianprocess.org/gpml/>.
- Greg Ridgeway. *gbm: Generalized Boosted Regression Models*, 2017. URL <https://CRAN.R-project.org/package=gbm>. R package version 2.1.3.
- AG Røseth, J Kristinsson, MK Fagerhol, H Schjønby, E Aadland, K Nygaard, and B Roald. Faecal calprotectin: a novel test for the diagnosis of colorectal cancer? *Scandinavian Journal of Gastroenterology*, 28(12):1073–1076, 1993.
- Olivier Roustant, David Ginsbourger, and Yves Deville. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based meta-modeling and optimization. *Journal of Statistical Software*, 51(1):1–55, 2012. URL <http://www.jstatsoft.org/v51/i01/>.
- Walter Rudin. *Real and Complex Analysis*. McGraw-Hill Book Company, third edition, 1987.
- Jssai Schur. Bemerkungen zur theorie der beschränkten bilinearformen mit unendlich vielen veränderlichen. *Journal für die Reine und Angewandte Mathematik*, 140:1–28, 1911.
- Gideon Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

- Dino Sejdinovic and Arthur Gretton. What is an RKHS? http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/RKHS_Notes1.pdf, 2012.
- CL Silva, M Passos, and JS Camara. Investigation of urinary volatile organic metabolites as potential cancer biomarkers by solid-phase microextraction in combination with gas chromatography-mass spectrometry. *British Journal of Cancer*, 105(12):1894, 2011.
- Johan Skog, Tom Würdinger, Sjoerd van Rijn, Dimphna H Meijer, Laura Gainche, William T Curry, Bob S Carter, Anna M Krichevsky, and Xandra O Breakefield. Glioblastoma microvesicles transport RNA and proteins that promote tumour growth and provide diagnostic biomarkers. *Nature Cell Biology*, 10(12):1470–1476, 2008.
- Robert A Smith, Vilma Cokkinides, Durado Brooks, Debbie Saslow, and Otis W Brawley. Cancer screening in the United States, 2010: a review of current American Cancer Society guidelines and issues in cancer screening. *CA: A Cancer Journal for Clinicians*, 60(2):99–119, 2010.
- Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, Lake Tahoe, USA, 2012.
- Helena A Soini, Iveta Klouckova, Donald Wiesler, Elisabeth Oberzaucher, Karl Grammer, Sarah J Dixon, Yun Xu, Richard G Brereton, Dustin J Penn, and Milos V Novotny. Analysis of volatile organic compounds in human saliva by a static sorptive extraction method and gas chromatography-mass spectrometry. *Journal of Chemical Ecology*, 36(9):1035–1042, 2010.
- Christian Steinruecken, Emma Smith, David Janz, James Lloyd, and Zoubin Ghahramani. The Automatic Statistician. In Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors, *Automated Machine Learning*, Series on Challenges in Machine Learning. Springer, May 2019. ISBN 978-3-030-05317-8. doi: 10.1007/978-3-030-05318-5_9.
- Charlene L Stokamer, Craig T Tenner, Jhuma Chaudhuri, Eva Vazquez, and Edmund J Bini. Randomized controlled trial of the impact of intensive patient education on compliance with fecal occult blood testing. *Journal of General Internal Medicine*, 20(3):278–282, 2005.
- George PH Styan. Hadamard products and multivariate statistical analysis. *Linear Algebra and its Applications*, 6:217–240, 1973.

- Masashi Sugiyama. Local Fisher discriminant analysis for supervised dimensionality reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 905–912, Pittsburg, Pa., USA, 2006. ACM.
- Christopher B Summerton, Michael G Longlands, Keith Wiener, and David R Shreeve. Faecal calprotectin: a marker of inflammation throughout the intestinal tract. *European Journal of Gastroenterology & Hepatology*, 14(8):841–845, 2002.
- Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
- Vladimir Svetnik, Andy Liaw, Christopher Tong, and Ting Wang. Application of Breiman’s random forest to modeling structure-activity relationships of pharmaceutical molecules. In *International Workshop on Multiple Classifier Systems*, pages 334–343, Cagliari, Italy, 2004. Springer.
- Cheryl Swanson. Are Jeff Bezos and Bill Gates wrong about Illumina? <https://www.fool.com/investing/general/2016/05/29/are-jeff-bezos-and-bill-gates-wrong-about-illumina.aspx>, May 2016. Date accessed: 2nd December 2018.
- Tcl Core Team. Tcl developer site. <https://www.tcl.tk/>, 2018. Accessed: 2018-09-07.
- Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. AutoWEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 847–855, Chicago, Ill., USA, 2013. ACM.
- J Tibble, G Sigthorsson, R Foster, R Sherwood, M Fagerhol, and I Bjarnason. Faecal calprotectin and faecal occult blood tests in the diagnosis of colorectal carcinoma and adenoma. *Gut*, 49(3):402–408, 2001.
- Michael E Tipping. Sparse kernel principal component analysis. In *Advances in Neural Information Processing Systems*, pages 633–639, Vancouver, British Columbia, Canada, 2001.

- Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.
- Michalis Titsias and Neil D Lawrence. Bayesian gaussian process latent variable model. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 844–851, Sardinia, Italy, 2010.
- Dustin Tran, Rajesh Ranganath, and David M Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.
- Florence E Turrentine, Chaderick E Denlinger, Virginia B Simpson, Robert A Garwood, Stephanie Guerlain, Abhinav Agrawal, Charles M Friel, Damien J LaPar, George J Stukenborg, and R Scott Jones. Morbidity, mortality, cost, and survival estimates of gastrointestinal anastomotic leaks. *Journal of the American College of Surgeons*, 220(2):195–206, 2015.
- Raquel Urtasun and Trevor Darrell. Discriminative Gaussian process latent variable model for classification. *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 927–934, 2007. doi: 10.1145/1273496.1273613. URL [citeulike-article-id:6539604](http://dx.doi.org/10.1145/1273496.1273613)<http://dx.doi.org/10.1145/1273496.1273613><http://portal.acm.org/citation.cfm?id=1273613>.
- Lesley Uttley, Becky L Whiteman, Helen Buckley Woods, Susan Harnan, Sian Taylor Philips, Ian A Cree, Early Cancer Detection Consortium, et al. Building the evidence base of blood-based biomarkers for early detection of cancer: a rapid systematic mapping review. *EBioMedicine*, 10:164–173, 2016.
- Andreas Wehinger, Alex Schmid, Sergei Mechtcheriakov, Maximilian Ledochowski, Christoph Grabmer, Guenther A Gastl, and Anton Amann. Lung cancer detection by proton transfer reaction mass-spectrometric analysis of human breath gas. *International Journal of Mass Spectrometry*, 265(1):49–59, 2007.
- Holger Wendland. *Scattered Data Approximation*, volume 17. Cambridge University Press, 2004.
- E Westenbrink, Ramesh P Arasaradnam, Nicola O’Connell, C Bailey, C Nwokolo, Karna Dev Bardhan, and JA Covington. Development and application of a new electronic nose instrument for the detection of colorectal cancer. *Biosensors and Bioelectronics*, 67:733–738, 2015.

- Michael Westhoff, Patric Litterst, Lutz Freitag, Wolfgang Urfer, Sabine Bader, and Joerg Ingo Baumbach. Ion mobility spectrometry for the detection of volatile organic compounds in exhaled breath of lung cancer patients. *Thorax*, 2009.
- Hadley Wickham, Jim Hester, Kirill Müller, and Daniel Cook. *memoise: Memoisation of Functions*, 2017. URL <https://CRAN.R-project.org/package=memoise>. R package version 1.1.0.
- MM Widlak, CL Thomas, MG Thomas, C Tomkins, S Smith, N O’connell, S Wurie, L Burns, C Harmston, C Evans, et al. Diagnostic accuracy of faecal biomarkers in detecting colorectal cancer and adenoma in symptomatic patients. *Alimentary Pharmacology & Therapeutics*, 45(2):354–363, 2017.
- Monika M Widlak, Matthew Neal, Emma Daulton, Claire L Thomas, Claudia Tomkins, Baljit Singh, Christopher Harmston, Alfian Wicaksono, Charles Evans, Steve Smith, et al. Risk stratification of symptomatic patients suspected of colorectal cancer using faecal and urinary markers. *Colorectal Disease*, 2018.
- Christopher KI Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems*, pages 295–301, Denver, Colo., USA, 1997.
- Christopher KI Williams and David Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.
- Peter M Williams. Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, 7(1):117–143, 1995.
- Ofer Yossepowitch, Harry W Herr, and S Machele Donat. Use of urinary biomarkers for bladder cancer surveillance: patient perspectives. *The Journal of Urology*, 177(4):1277–1282, 2007.
- Ru Zhang, C. Devon Lin, and Pritam Ranjan. Local gaussian process model for large-scale dynamic computer experiments. *Journal of Computational and Graphical Statistics*, 27(4):798–807, 2018. doi: 10.1080/10618600.2018.1473778. URL <https://doi.org/10.1080/10618600.2018.1473778>.